LHC COMPUTING GRID

MANUALS SERIES

LCG-2 USER GUIDE

Document identifier:	**CERN-LCG-GDEIS-454439**
EDMS id:	**454439**
Version:	**v1.1**
Date:	**March 26, 2004**
Section:	**LCG Experiment Integration and Support**
Document status:	**DRAFT**
Author(s):	Antonio Delgado Peris, Patricia Méndez Lorenzo, Flavia Donno, Andrea Sciabà, Simone Campana, Roberto Santinelli
File:	**LCG-2-Userguide**

Abstract: *This guide is an introduction to the LCG-2 Grid from a user's point of view*

Document Change Record

Issue	Item	Reason for Change
09/03/04	v1.0	First Draft
17/03/04	v1.1	Corrections from EIS group comments

Files

Software Products	User files
PDF	LCG-2-UserGuide.pdf
PS	LCG-2-UserGuide.ps

CONTENTS

1 INTRODUCTION 7

1.1 OBJECTIVES OF THIS DOCUMENT 7

1.2 APPLICATION AREA 7

1.3 DOCUMENT EVOLUTION PROCEDURE 7

1.4 APPLICABLE DOCUMENTS AND REFERENCE DOCUMENTS 7

1.5 TERMINOLOGY 9

1.5.1 Glossary 9

2 EXECUTIVE SUMMARY 12

3 OVERVIEW 13

3.1 THE LCG-2 ARCHITECTURE 13

3.1.1 Getting Started 14

3.1.2 The User Interface 14

3.1.3 Computing Element and Storage Element 15

3.1.4 Information System 15

3.1.5 Data Management 17

3.1.6 Job Management 19

3.2 SERVICE INTERACTIONS AND JOB FLOW 19

3.2.1 Job Submission 20

3.2.2 Data Management 21

3.2.3 Information System 22

4 GETTING STARTED 24

4.1 OBTAINING A CERTIFICATE 24

4.2 REGISTERING WITH LCG-2 25

4.3 VIRTUAL ORGANIZATIONS 26

4.4 SETTING UP THE USER ACCOUNT 26

4.5 CHECKING A CERTIFICATE 27

4.6 PROXY CERTIFICATES 28

4.6.1 Virtual Organization Membership Service 30

4.7 ADVANCED PROXY MANAGEMENT 31

5 JOB MANAGEMENT 34

5.1 THE COMMAND LINE INTERFACE 34

5.1.1 Job Submission 34

5.1.2 Job Description Language 37

5.1.3 Job Operations 41

5.1.4 Checkpointable Jobs 44

5.1.5 Interactive Jobs 45

5.1.6 MPI Jobs 45

5.1.7 Advanced Command Options 46

5.1.8 The BrokerInfo 47

5.2 THE GRAPHICAL USER INTERFACE 48

6 DATA MANAGEMENT 49
6.1 INTRODUCTION 49
6.1.1 EDG Data Management Tools 49
6.1.2 File Names within LCG-2 49
6.2 EDG-REPLICA-MANAGER CLIENT TOOLS 50
6.2.1 Basic Replica Manager Commands 51
6.2.2 Other Commands 55
6.2.3 Accessing a Grid File from a Job 56
6.3 EDG-LRC AND EDG-RMC CLIENT TOOLS 59
6.3.1 Local Replica Catalog Commands 59
6.3.2 Replica Metadata Catalog Commands 62
6.4 LOW LEVEL DATA MANAGEMENT TOOLS 64
6.5 POOL AND LCG-2 65
6.5.1 LCG Catalog (RLS) vs POOL Catalog (XML) 65
7 INFORMATION SYSTEM 67
7.1 THE LOCAL GRIS 67
7.2 THE SITE GIIS 69
7.3 THE BDII 72
A THE GLUE SCHEMA 74
A.1 THE GLUE SCHEMA LDAP TREE 74
A.2 ATTRIBUTES FOR THE COMPUTING ELEMENT 77
A.3 ATTRIBUTES FOR THE STORAGE ELEMENT 81

A.4 ATTRIBUTES FOR THE CE-SE BINDING 83

B THE GRID MIDDLEWARE 84

C JOB STATUS DEFINITION 85

1. INTRODUCTION

1.1. OBJECTIVES OF THIS DOCUMENT

This document gives an overview of the main services of the LCG-2 facility. It allows users to understand the building blocks and the available interfaces to the GRID tools in order to run jobs and manage data.

This document is not an Administration Guide.

1.2. APPLICATION AREA

This guide is addressed to users and site administrators of the LCG-2 facility who would like to work on the LCG-2 service.

1.3. DOCUMENT EVOLUTION PROCEDURE

This document updates the previous LCG-1 User Guide ([R1]).

The guide reflects the current status of the LCG-2 service, and will be modified accordingly with the new LCG-2 releases. In some points of the document, references to the foreseeable future of the LCG-2 service are made.

1.4. APPLICABLE DOCUMENTS AND REFERENCE DOCUMENTS

APPLICABLE DOCUMENTS

[A1] EDG User's Guide
http://marianne.in2p3.fr/datagrid/documentation/EDG-Users-Guide-2.0.pdf

[A2] LDAP Services User Guide
http://hepunx.rl.ac.uk/edg/wp3/documentation/wp3-ldap_user_guide.html

[A3] LCG-1 User Scenario
https://edms.cern.ch/document/414211/

REFERENCES

[R1] LCG-1 User Guide
http://grid-deployment.web.cern.ch/grid-deployment/eis/docs/LCG-1-UserGuide.htm

[R2] Regional Centres for LHC computing
The MONARC Architecture Group
http://barone.home.cern.ch/barone/monarc/RCArchitecture.html
http://monarc.web.cern.ch/MONARC/

[R3] The Anatomy of the Grid.
Enabling Scalable Virtual Organizations
Ian Foster, Carl Kesselman, Steven Tuecke
http://www.globus.org/research/papers/anatomy.pdf

[R4] Overview of the Grid Security Infrastructure
http://www-unix.globus.org/security/overview.html

[R5] Resource Management
http://www-unix.globus.org/developer/resource-management.html

[R6] WP1 Workload Management Software – Administrator and User Guide. Nov 24th, 2003
http://server11.infn.it/workload-grid/docs/DataGrid-01-TEN-0118-1_2.pdf

[R7] The GridFTP Protocol and Software
http://www.globus.org/datagrid/gridftp.html

[R8] MDS 2.2 Features in the Globus Toolkit 2.2 Release
http://www.globus.org/mds/

[R9] European DataGrid Project
http://eu-datagrid.web.cern.ch/eu-datagrid/

[R10] The GLUE schema
http://www.cnaf.infn.it/~sergio/datatag/glue/

[R11] LCG-2 Manual Installation Guide
https://edms.cern.ch/file/434070//LCG2Install.pdf

[R12] Classified Advertisements. Condor.
http://www.cs.wisc.edu/condor/classad

[R13] The Condor Project.
http://www.cs.wisc.edu/condor/

[R14] Job Description language HowTo. December 17th, 2001
http://server11.infn.it/workload-grid/docs/DataGrid-01-TEN-0102-0_2-Document.pdf

[R15] JDL Attributes – Release 2.x. Oct 28th, 2003
http://server11.infn.it/workload-grid/docs/DataGrid-01-TEN-0142-0_2.pdf

[R16] WP1 Workload Management System – Job Partitioning and Checkpointing. June 3, 2002
https://edms.cern.ch/file/347730/1/DataGrid-01-TED-0119-0_3.pdf

[R17] The EDG-Brokerinfo User Guide - Release 2.x. 6th August 2003
http://server11.infn.it/workload-grid/docs/edg-brokerinfo-user-guide-v2_2.pdf

[R18] Workload Management Software – GUI User Guide. Nov 24th, 2003
http://server11.infn.it/workload-grid/docs/DataGrid-01-TEN-0143-0_0.pdf

[R19] User Guide for the EDG Replica Manager 1.5.x
http://edg-wp2.web.cern.ch/edg-wp2/replication/docu/r2.1/edg-replica-manager-userguide.pdf

[R20] User Guide for the EDG Local Replica Catalog 2.1.x
http://edg-wp2.web.cern.ch/edg-wp2/replication/docu/r2.1/edg-lrc-userguide.pdf

[R21] User Guide for the EDG Replica Metadata Catalog 2.1.x
http://edg-wp2.web.cern.ch/edg-wp2/replication/docu/r2.1/edg-rmc-userguide.pdf

[R22] EDG Tutorial – Handout for Participants for EDG Release 2.x
http://edms.cern.ch/document/393671

[R23] User Guide for the EDG Replica Optimization Service 2.1.x
http://edg-wp2.web.cern.ch/edg-wp2/replication/docu/r2.1/edg-ros-userguide.pdf

[R24] User Guide for the Replica Location Index 2.1.x
http://edg-wp2.web.cern.ch/edg-wp2/replication/docu/r2.1/edg-rli-userguide.pdf

[R25] Developer Guide for the EDG Replica Manager 1.5.x
http://edg-wp2.web.cern.ch/edg-wp2/replication/docu/r2.1/edg-replica-manager-devguide.pdf

[R26] Remote File Stream. Extensions to the Standard C++ I/O Library for Accessing Remote Files
http://doc.in2p3.fr/doc/public/products/rfstream/rfstream.html

[R27] POOL - Persistency Framework. Pool Of persistent Objects for LHC.
http://lcgapp.cern.ch/project/persist
Learning POOL by examples, a mini tutorial.
http://lcgapp.cern.ch/project/persist/tutorial/learningPoolByExamples.html

1.5. TERMINOLOGY

1.5.1. Glossary

API: Application Programming Interface

BDII: Berkeley Database Information Index
CE: Computing Element
CERN: European Laboratory for Particle Physics
ClassAd: Classified advertisement
CLI: Command Line Interface
CNAF: INFN's National Center for Telematics and Informatics
DIT: Directory Information Tree
DN: Distinguished Name (LDAP's)
EDG: European DataGrid
EDT: European DataTag
FNAL: Fermi National Accelerator Laboratory
GIIS: Grid Index Information Server
GLUE: Grid Laboratory for a Uniform Environment
GRAM: Globus Resource Allocation Manager
GRIS: Grid Resource Information Service
GSI: Grid Security Infrastructure
GUI: Graphical User Interface
GUID: Grid Unique ID
ID: Identifier
INFN: Instituto Nazionale di Fisica Nucleare
IS: Information Service
JCS: Job Control Service
JDL: Job Description Language
LB: Logging and Bookkeeping Service
LDAP: Lightweight Directory Access Protocol
LFN: Local File Name
LRC: Local Replica Catalog
LRMS: Local Resource Management System
LSF: Load Sharing Facility
MDS: Monitoring and Discovery Service
MPI: Message Passing Interface
MSS: Mass Storage System
NS: Network Server
OS: Operating System
PBS: Portable Batch System
PFN: Physical File name
PID: Process IDentifier
POOL: Pool of Persistent Objects for LHC
RAL: Rutherford Appleton Laboratory
RB: Resource Broker
RLI: Replica Location Index
RLS: Replica Location Service
RM: Replica Manager
RMC: Replica Metadata Catalog

RMS: Replica Management System
ROS: Replica Optimization Service
SASL: Simple Authorization & Security Layer (LDAP)
SE: Storage Element
SMP: Symmetric Multi Processor
SRM: Storage Resource Manager
SURL: Storage URL
TURL: Transport URL
UI: User Interface
URI: Uniform Resource Identifier
URL: Universal Resource Locator
UUID: Universal Unique ID
VDT: Virtual Data Toolkit
VO: Virtual Organisation
WMS: Workload Management System
WN: Worker Node
WPn: Work Package #n

2. EXECUTIVE SUMMARY

This user guide is intended for users of the LCG-2 service. Within these pages, the user will hopefully find an adequate introduction to the services provided by the Grid and a description of how to use them. Examples are given for the management of jobs and data, the monitoring of resources status, etc., in order to easily be effective.

A first introduction on the organization of the service itself is presented in Chapter 3. The reader can skip this chapter if he/she is familiar already with the basic architecture of the LCG-2 service. In Chapter 4, the procedures to register with LCG, get a certificate and manage proxies are described.

An overview of the Workload Management service is given in Chapter 5. It explains the basic commands for job submission and management, as well as those for retrieving information related to the Workload Management match-making mechanism from inside a Grid job.

Data Management services are described in Chapter 6. Not only the high-level interface is described but also commands that can be useful in case of problems or for debugging purposes.

Details on how to find out about the status of LCG-2 resources are given in Chapter 7, where the Information System is discussed. Many examples are provided to interrogate GRISes, the LCG-2 top GIISes, and the BDII.

Finally, in the appendices, details about the Glue Schema used to describe LCG-2 resources (Appendix A), the version of the middleware and the components used (Appendix B) and a description of the evolution of the job status during submission and execution (Appendix C) are given.

3. OVERVIEW

The job of the LHC computing Grid Project –LCG– is to prepare the computing infrastructure for the simulation, processing and analysis of LHC data for all four of the LHC collaborations: ***ALICE, ATLAS, CMS*** and ***LHCB***. This includes both the common infrastructure of libraries, tools and frameworks required to support the physics application software, and the development and deployment of the computing services needed to store and process data, providing batch and interactive facilities for the worldwide community of physics involved in LHC.

The requirements for LHC data handling are very large, in terms of computational power, data storage capacity, data access performance and the associated human resources for operation and support. It is not considered feasible to fund all of the resources at one site, and so it has been agreed that the LCG computing service will be implemented as a geographically distributed ***Computational Data Grid***. This means that the service will use computing and storage resources, installed at a large number of ***Regional Computing Centres*** in many different countries, interconnected by fast networks. Special software, referred to generically as ***Grid Middleware***, will hide much of the complexity of this environment from the user, giving the impression that all of these resources are available in a coherent virtual computer centre.

In the first phase of the project, from 2002 through 2005, LCG will develop and prototype the computing services and deploy a series of computing data challenges of increasing size and complexity to demonstrate the effectiveness of the software and computing models selected by the experiments.

LCG-2 is the new release of LCG (after LCG-1). This new version will be running in 2004 and its main goal is to provide a stable service. LCG-2 expands the services of LCG-1, with enough resources and functionality for the *2004 Computing Data Challenge*. In addition, more Tier 1 and Tier 2 centres will join the project, following the Monarc model [R2], as in the previous LCG-1 release.

In the first phase of LCG-2, the core sites implementing the new release are CERN, Karlsruhe, Barcelona, FNAL, CNAF, Nikhef, Taipei and RAL.

LCG-2 is organized into Virtual Organizations [R3]: dynamic collections of individuals and institutions sharing resources in a flexible, secure and coordinated manner. In such settings, we encounter unique authentication, authorization, resource access, resource discovery, and other challenges.

3.1. THE LCG-2 ARCHITECTURE

This section provides an overview of the LCG-2 architecture.

3.1.1. Getting Started

Before LCG resources can be used, a user is required to register some personal data and information about the Virtual Organization he/she belongs to with the ***LCG Registration Server***. CERN will run such a service, collecting information about all LCG users.

The ***Grid Security Infrastructure (GSI)*** in LCG-2 enables secure authentication and communication over an open network [R4]. GSI is based on public key encryption, X.509 certificates, and the Secure Sockets Layer (SSL) communication protocol. Extensions to these standards have been added for single sign-on and delegation.

In order to access Grid resources, a user needs to have a digital X509 certificate from a ***Certification Authority (CA)*** recognized by LCG. The CAs recognized by LCG are listed later on.

In LCG-1 there were five possible Virtual Organizations (VOs) a user could be affiliated to: one for the DTeam (LCG Grid Deployment Group) and one more for each one of the four LCG HEP experiments. A ***Virtual Organization Server*** maps users certificates to users data and lists the certificates as belonging to users that are part of a VO or HEP experiment. The VO Server for the DTeam was and still is run at CERN, while the VO Servers for the four experiments are run at NIKHEF. But LCG-2 can support many more VOs. Each site installing the new release is free to support any VOs. Users can be aware of the VOs supported at a given site by asking directly to that site. The commands used for that purpose are shown later.

A user is authorized to use LCG-2 Grid resources by means of the ***grid-mapfile*** mechanism. Each host part of the LCG-2 Grid has a local grid-mapfile which maps user certificates to local accounts. When a user request-for-service reaches a host, the certificate of the user is checked in the local grid-mapfile. If the user certificate is found there, then the local account to which the user certificate is mapped is used to serve the request. The same is true for services. Details are explained in [R4].

The following sections describe several types of services run in LCG-2 to provide the Grid functionality.

3.1.2. The User Interface

The initial point of access to the LCG-2 Grid is the ***User Interface (UI)***. This is a machine where LCG users have a personal account and where the user's certificate is installed. This is the gateway to Grid services. From the UI, a user can be authenticated and authorized to use the LCG-2 Grid resources. This is the component that allows users to access the functionalities offered by the Information, Workload and Data management services. It provides a Command Line Interface (CLI) to perform some basic grid operations:

- submit a job for execution on a Computing Element;

- list all the resources suitable to execute a given job;
- replicate and copy files;
- cancel one or more jobs;
- retrieve the output of one or more finished jobs;
- show the status of one or more submitted jobs.

One or more UIs are available at each site part of the LCG-2 Grid.

3.1.3. Computing Element and Storage Element

A ***Computing Element (CE)*** is defined as a Grid batch queue and it is identified by a pair *<hostname>/<batch_queue_name>*. *A Computing Element is a homogeneous farm of computing nodes called* ***Worker Nodes (WN)*** *and a node acting as a* ***Grid Gate (GG)*** *or front-end to the rest of the Grid. The GG runs a Globus gatekeeper, the Globus GRAM (Globus Resource Allocation Manager) [R5], the master server of a Local Resource Managemente System (LRMS), together with the EDG Logging and Bookkeeping services [R6]. In LCG-2 the types of LRMS supported are PBS, LSF and Condor. While all WNs can be hidden and running behind a firewall, the Gate node must be accessible from outside the site. The GG is responsible for accepting jobs and dispatching them for execution to the WNs. The GG provides a uniform interface to the computational resources it manages. On the WNs, all commands and Application Programming Interface (API) for performing actions on Grid resources and Grid data are available.*

Each LCG-2 site runs at least one CE and a farm of WNs behind it.

A ***Storage Element (SE)*** *provides uniform access and services to large storage spaces. The Storage Element may control large disk arrays, mass storage systems (MSS) and the like. The current LCG-2 release includes a classic SE, which has a* ***GridFTP*** *server [R7] as the data transfer protocol to the storage resource. It is responsible for secure, fast and efficient file transfer to/from the Storage Element.*

In the final LCG-2 release, though, this storage resource will be managed by a ***Storage Resource Manager (SRM)****. This middleware module, will make it possible to dinamically manage the contents of the storage resource at any time. The SRM will interact with the operating system, with the mass storage system (to perform file archiving), and with the protocols (to perform file transfer operations).*

As MSS, LCG-2 will support disk pool (with GridFTP and rfio as transfer protocols), tape archiving systems (with GridFTP and rfio) and nstore (with GridFTP). The file protocol is no longer supported in LCG-2.

Each LCG-2 site provides one or more SEs.

3.1.4. Information System

The resources described up to now constitute the compute and storage power of the LCG-2 Grid. Together with that infrastructure, additional services are provided to locate and report on the status of Grid resources, to find the

*most appropriate resources to run a job requiring certain data access and to automatically perform data operations necessary before and after a job is run. These are the **Information System** and the **Data Management** services.*

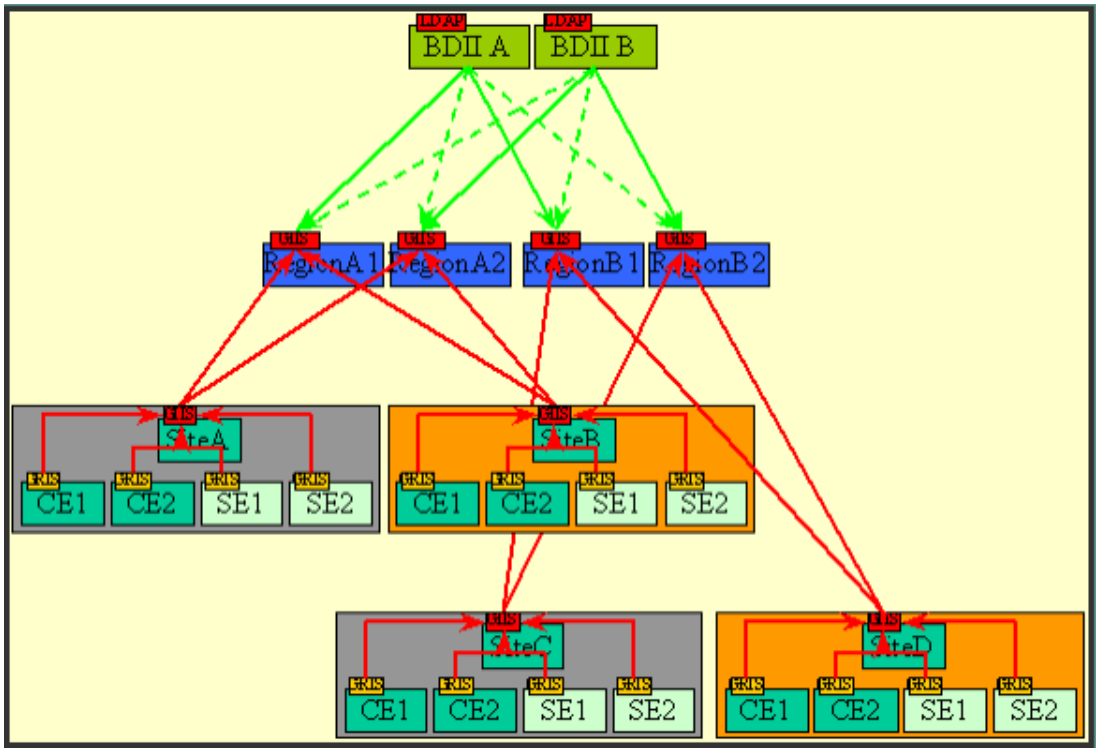


Figure 1: The Information System in LCG-2

*The Information System (IS) provides information about the LCG-2 Grid resources and their status. In LCG-2, the **Monitoring and Discovery Service (MDS)** from Globus [R8] has been adopted as the provider of this service.*

*Figure 1 shows how the information is stored and propagated. Information is propagated in a hierarchy: Compute and storage resources at a site report (via the **Grid Resource Information Servers, or GRISes**) their static and dynamic status to the **Site Grid Index Information Server (GIIS)**. In LCG-2, the site GIISes register with one or more regional GIISes, for redundancy reasons and to nicely divide the administration domains. Figure 2 shows the sites hosting Regional GIISes. This structure is transparent to the user.*

*Due to dynamic nature of the GRID, the GIISes might not contain information about resources that are actually available on the Grid but that, for some reasons, are unable to publish updated information to the GIISes. Because of this, the **Berkeley DB Information Index (BDII)** was introduced. The BDII queries the regional GIISes and acts as a cache storing information about the Grid status in its database. Every time a resource appears in one of the GIISes, its existence is registered in one of the BDIIs. There is one BDII running at each site where a Resource Broker (see later) is installed. Users and other Grid services (such as the RB) can interrogate BDIIs to get information about the Grid status. Very up-to-date information can be found by directly interrogating the site GIISes or the local GRISes that run on the specific resources. Later on we describe how a user can interrogate these services.*

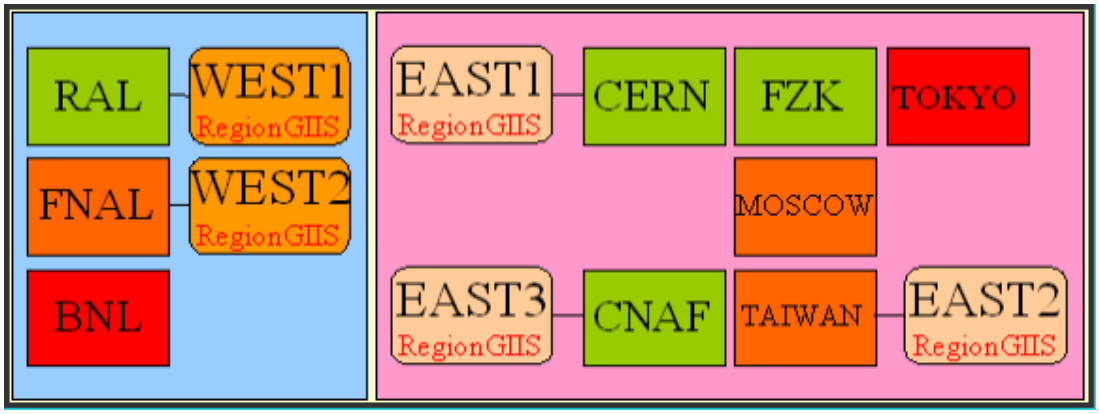


Figure 2: Sites hosting GIISes in LCG-2

3.1.5. Data Management

*The Data Management services are provided by the **Replica Management System (RMS)** of the **European DataGrid (EDG)** [R9]. In a Grid environment, the data files are replicated, possibly on a temporary basis, to many different sites depending on where the data is needed. The users or applications do not need to know where the data is located. They use logical names for the files and the Data Management services are responsible for locating and accessing the data.*

*The files in the Grid are referenced by different names: **Grid Unique IDentifier (GUID)**, **Logical File Name (LFN)**, **Storage URL (SURL)** and **Transport URL (TURL)**. While the GUID or LFN refer to files and not replicas, and say nothing about locations, the SURLs and TURLs give information about where a physical replica is located.*

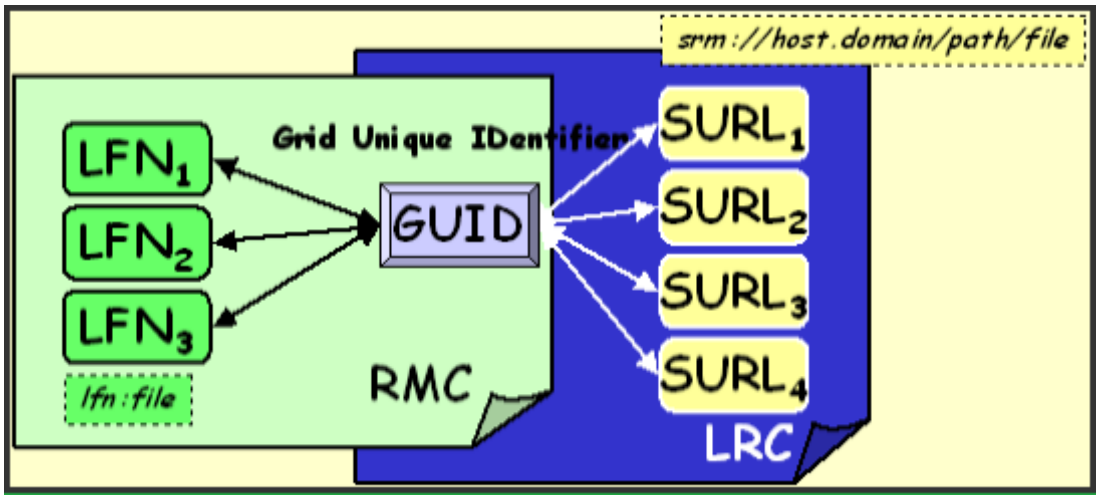


Figure 3: Different filenames in LCG-2

A file can always be identified by its GUID; this is assigned at data registration time and is based on the UUID standard to guarantee unique IDs. A GUID is of the form: guid:<unique_string>. All the replicas of a file will share the same GUID. In order to locate a Grid accessible file, the human user will normally use a LFN. LFNs are usually more intuitive, human-readable strings, since they are allocated by the user as GUID aliases. Their form is: lfn:<any_alias>.

The SURL is used by the RMS to find where a replica is physically stored, and by the SE to locate it. Currently, the SURLs are of the form: sfn:<SE_hostname>/<local_string>[1]*, where <local_string> is used internally by the SE to locate the file.*

Finally, the TURL gives the necessary information to retrieve a physical replica, including hostname, path, protocol and port (as any conventional URL); so that the application can open and retrieve it. Figure 3 shows the relation between the different file names.

*The main services offered by the RMS are: the **Replica Location Service (RLS)** and the **Replica Metadata Catalog (RMC)**.*

The RLS maintains information about the physical location of the replicas (mapping with the GUIDs). It is composed of Local Replica Catalogs (LRCs) which hold the information of replicas for a single VO.

The RMC stores the mapping between GUIDs and the respective aliases (LFNs) associated with them, and maintains other metada information (sizes, dates, ownerships...)

*The last component of the Data Management framework is the **Replica Manager**. The Replica Manager presents a single interface for the RMS to the user, and interacts with the other services. This is illustrated in Figure 4. In the LCG-2, this interface is integrated with the User Interface described earlier.*

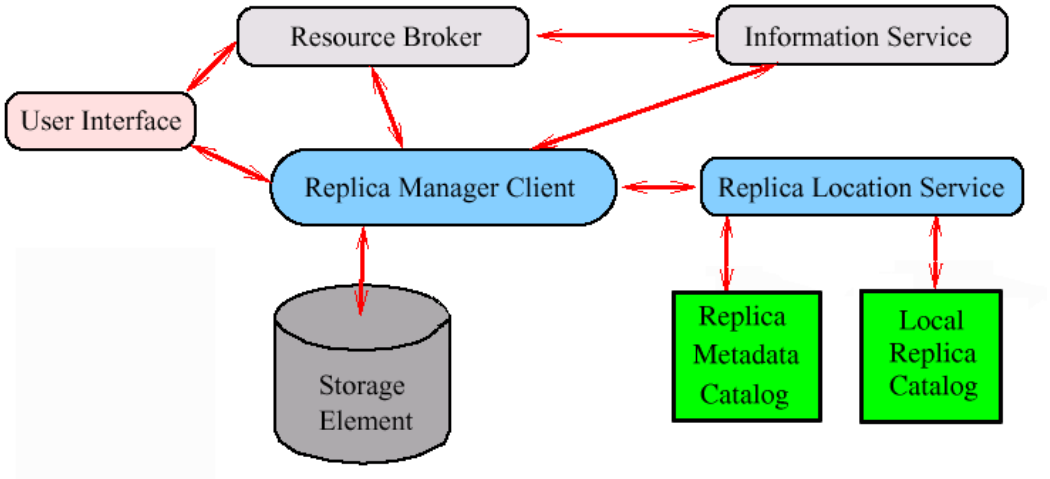


Figure 4: Interactions of the Replica Manager with other grid components

[1]When SRMs are already working, files stored there will use *srm as the prefix for their SURLs, instead of sfn. This distinction will allow the RMS to distinguish which kind of storage the file is on.*

For the moment these catalogues are centralized and there is one RLS (with its LRC and RMC) per VO. In the first phase, all RLSs are run at CERN.

3.1.6. Job Management

*The services of the **Workload Management System (WMS)** are responsible for the acceptance of job submits and the dispatching of those jobs to the appropiate CE, depending on the job requirements and the available resources. For that purpose, it must retrieve information from the BDII, and the RLS. The **Resource Broker (RB)** is the machine where the services of the WMS run. These services are:*

- ***Network Server (NS)**, which accepts the incoming job requests from the UI, and provides support for the job control functionality.*
- ***Workload Manager**, which is the core component of the system.*
- ***Match-Maker** (also called Resource Broker), whose duty is finding the best resource matching the requirements of a job (match-making process).*
- ***Job Adapter**, which prepares the environment for the job and its final description, before passing it to the Job Control Service.*
- ***Job Control Service (JCS)**, which finally performs the actual job management operations (job submission, removal...)*

*In addition, the **Logging and Bookkeeping service (LB)** is usually also run on a RB machine. The LB logs all job management Grid events, which can then be retrieved by users or system administrators for monitoring or troubleshooting.*

Multiple RBs are available in LCG-2 Grid. Participating sites are free to install their own RBs. To know which sites have installed an RB, the LCG-2 deployment status page can be consulted at:

http://grid-deployment.web.cern.ch/grid-deployment/cgi-bin/index.cgi?var=gis/lcg2Status

*The last component of the LCG-2 Grid described here is the **Proxy Server (PS)**. When a user accesses the Grid, he/she is provided with a temporary certificate, called proxy, that has an expiration time. If the user proxy expires before the user job has finished, all subsequent requests for service will fail due to unauthorized access. In order to avoid this, the Workload Management Service provided by EDG allows for proxy renewal before the expiration time has been reached if the job requires it. The PS is the component that allows such functionality.*

In LCG-2, a site is free to install a PS. Which sites have installed a PS can be consulted at the LCG-2 deployment status page indicated above.

Figure5 shows a summary of all LCG-2 service components available at CERN.

3.2. Service Interactions and Job Flow

This section describes briefly what happens when a user submits a job to the LCG-2 Grid to process some data and how the different components interact. We also give a description of the components of the Data Management

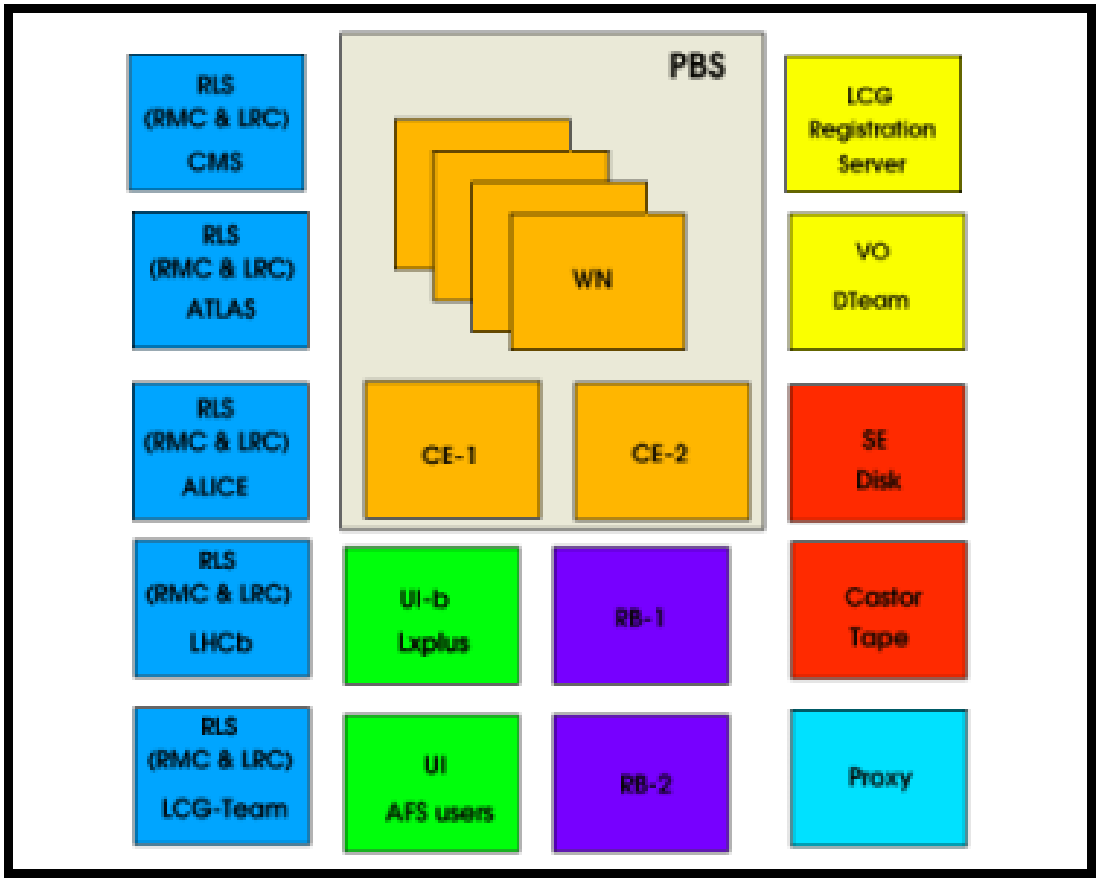


Figure 5: LCG-2 available services at CERN

system. User applications and further functionality can be built/developed on top of what is offered by LCG-2 Grid.

3.2.1. Job Submission

a. *After obtaining a digital certificate from one of the LCG-2 trusted Certification Authorities, registering with LCG-2, registering with a Virtual Organization and obtaining an account on an LCG-2 User Interface (once only actions), the user is ready to use LCG-2 Grid. He/she logs to the UI machine and creates a proxy certificate that authenticates him/her in every secure interaction, and has a limited lifetime.*

b. *The user submits the job from the UI to the WMS, where the job will be executed on a computing node. The user can specify in the job description file one or more files to be copied from the UI to the RB node; this set of files is called* ***Input Sandbox****. The event is logged in the LB and the status of the job is* ***SUBMITTED****.*

c. *The WMS, and in particular the Match-Maker component, looks for the best available CE to execute the job. To do so, the Match-Maker interrogates the BDII to query the status of computational and storage resources and the RLS to find location of data. The event is logged in the LB and the status of the job is* ***WAIT****.*

d. *The WMS Job Adapter prepares the job for submission creating a wrapper script that is passed, together with other parameters, to the JCS for submission to the selected CE. The event is logged in the LB and the status of the job is* ***READY****.*

e. *The Globus Gatekeeper on the CE receives the request and sends the Job for execution to the LRMS (e.g. PBS, LSF or Condor). The event is logged in the LB and the status of the job is* ***SCHEDULED****.*

f. *The LRMS handles the job execution on the available local farm worker nodes. User's files are copied from the RB to the WN where the job is executed. The event is logged in the LB and the status of the job is* ***RUNNING****.*

g. *While the job runs, Grid files can be accessed on the (close) SE using either the rfio protocol or local access if the files are copied to the WN local filesystem. In order for the job to find out which is the close SE, or what is the result of the Match-Maker process, a file with this information is produced by the WMS and shipped together with the job to the WN. This is known as the .BrokerInfo file. Information can be retrieved from this file using the BrokerInfo CLI or the API library.*

h. *The job can produce new output data that can be uploaded to the Grid and made available for other Grid users to use. This can be achieved using the Data Management tools described later. Uploading a file to the Grid means copying it on a Storage Element and registering its location, metadata and attribute to the RMS. At the same time, during job execution or from the User Interface, data files can be replicated between two SEs using again the Data Management tools.*

i. *If the job reaches the end without errors, the output (not large data files, but just small output files specified by the user in the so called* ***Output Sandbox****) is transferred back to the RB node. The event is logged in the LB and the status of the job is* ***DONE****.*

j. *At this point, the user can retrieve the output of his/her job from the UI using the WMS CLI or API. The event is logged in the LB and the status of the job is* ***CLEARED****.*

k. *Queries of the job status are addressed to the LB database from the UI machine. Also, from the UI is it possible to query the BDII for a status of the resources.*

l. *If the site where the job is being run falls down, the job will be automatically resent to another CE that is analogue to the previous one, and following the same requirements the user asked for. In the case that this new submission is disabled, the job will be marked as aborted. Users can get information about the scenario by simply questioning the LB service.*

Figure 6 shows what has been described in steps b to k.

3.2.2. Data Management

The Input/Output Sandbox is a mechanism for transferring small data files needed to start the job or to check the final status over the Grid. Large data files are available on the Grid and known to other users only if they are stored on SEs and registered in the RMS catalogues. In order to optimise data access and to introduce fault-tolerance and redundancy, data files can be replicated on the Grid. The EDG Replica Manager, the Replica Location Service and the Replica Metadata Catalog are the tools available for performing these tasks. Only anonymous access to the data catalogues is supported: the user proxy is not used to control the access to them.

In the LCG-2, as explained earlier, a file is identified uniquely by the GUID, but the user may refer to ir using different aliases. Also, there will probably be several physical replicas of each file. The user should never

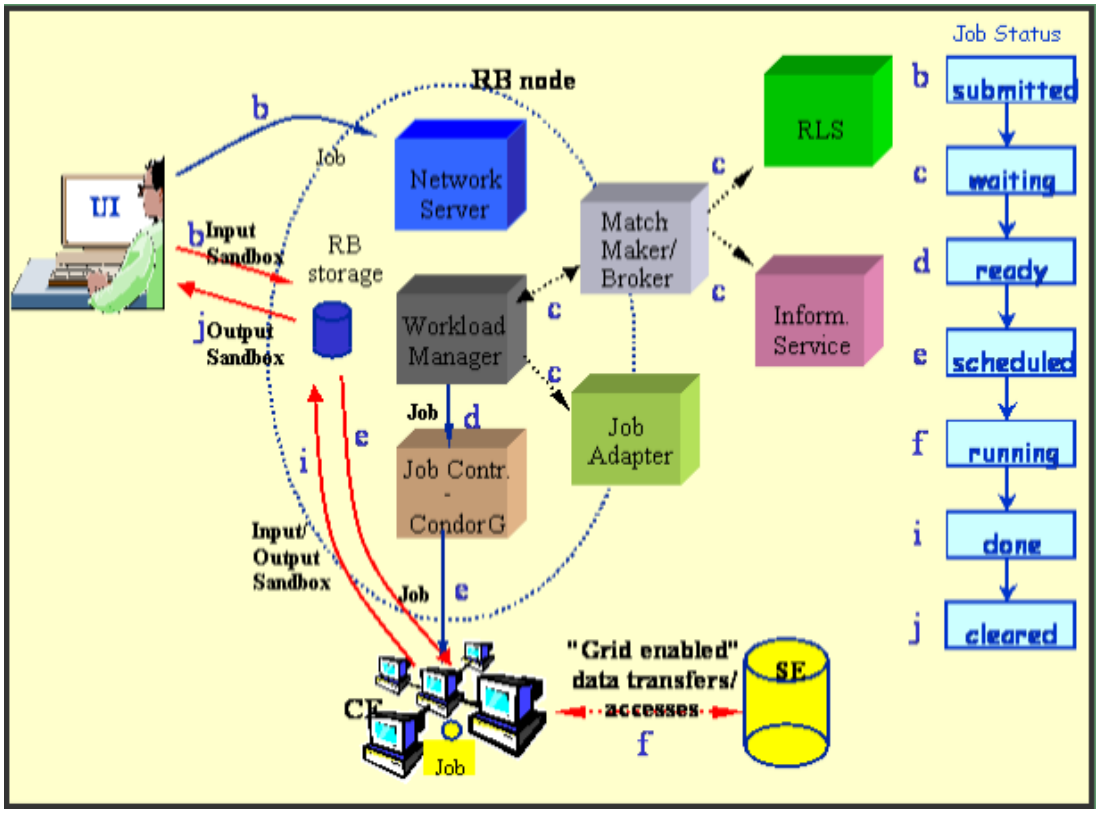


Figure 6: Job flow in the LCG-2

interact with the RMC or the RLS catalogsi directly. Instead, he/she should always use the EDG RM, or the POOL interface.

m. *When a new file is produced, the file should be uploaded to the Grid to be known and usable by Grid services or other Grid users. This can be done using the EDG Replica Manager commands for copying and registering a file.*

n. *Before running a job on the Grid, the user can ask the WMS to run the job on a CE close to an SE containing the data of interest, or, at run time, the job can ask the RMS to replicate a file on a SE close or even on the WN where the job is running.*

o. *If a file is no longer needed, it can be deleted from the Grid and all its references removed from the data catalogues.*

3.2.3. Information System

The architecture of the Information System in the LCG-2 Grid has been already described. Users can interrogate the IS to retrieve static or dynamic information about the status of the LCG-2. In order to have an optimal answer,

users are encouraged to query the BDIIs or the site GIISes, but not the regional GIISes. Also, the specific GRISes can be queried. Details and examples on how to interrogate GRIS, GIIS and BDII are given in Chapter 7.

The IS is based on OpenLDAP, an open source implementation of the Lightweight Directory Access Protocol (LDAP). LDAP is a protocol that provides the infrastructure for a directory service. A directory service is a specialized database optimized for reading, browsing and searching information. No transaction or roll-back features are normally offered. In particular in LCG-2 Grid, only anonymous access to the catalogue is offered. ***This means that all users can browse the catalogues and all services are allowed to enter information into it.***

The LDAP information model is based on entries. An entry is a collection of attributes which together form a globally unique ***Distinguished Name (DN)****, a name that uniquely identifies the entry. Each of the entry's attributes has a type and one or more values. The types are typically mnemonic strings, like "cn", while the syntax of the values depends on the attribute type. An* ***LDAP schema*** *describes the attributes and the types of the attributes associated with entries.*

Directory entries are arranged in a hierarchical tree-like structure referred to as Directory Information Tree (DIT) as shown in Figure 7.

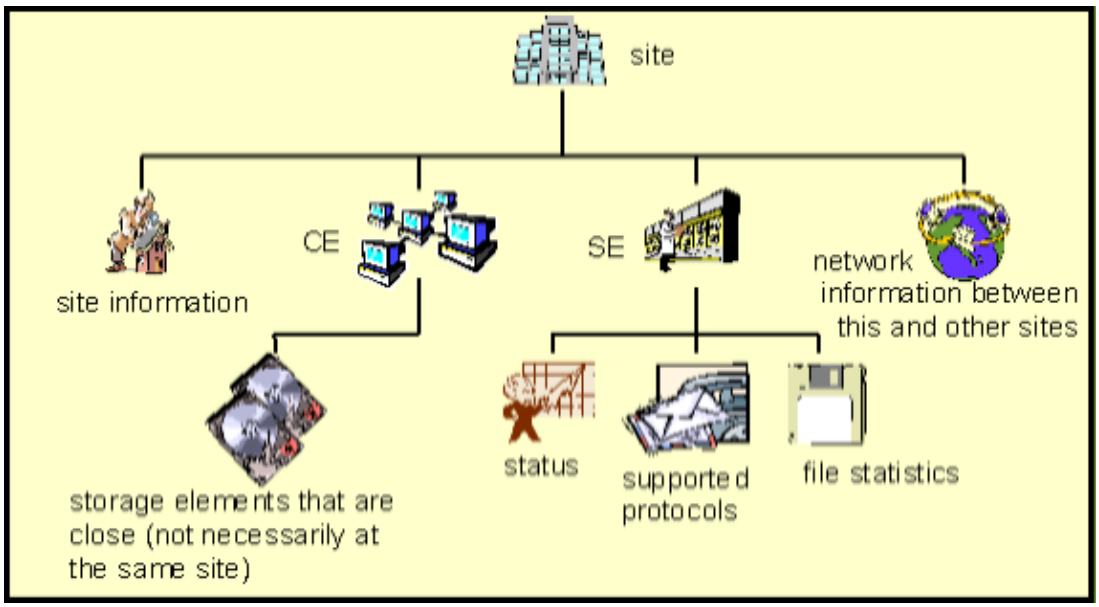


Figure 7: The Directory Information Tree (DIT)

The LCG-2 Grid deploys the ***GLUE (Grid Laboratory for a Uniform Environment) Schema*** *for information description. The GLUE Schema activity aims to define a common conceptual data model to be used for grid resources monitoring and discovery. There are three main components of the GLUE Schema. They describe the attributes and value of Computing Elements, Storage Elements and binding information for Computing and Storage Elements. Details can be found in [R10]. Examples on how to query the Information System in LCG-2 are given later on.*

4. GETTING STARTED

This section describes the preliminary steps to gain access to the LCG-2 Grid. Before using the LCG-2 Grid, the user must do the following:

1. *Obtain a Cryptographic X.509 certificate from an LCG-2 approved* ***Certification Authority (CA)****.*
2. *Get registered with LCG-2.*
3. *Join one of the LCG-2 Virtual Organizations.*
4. *Obtain an account on a machine which has the LCG-2 User Interface software installed.*
5. *Create a proxy certificate.*

Steps 1 to 4 need to be executed only once to have access to the Grid. Step 5 needs to be executed the first time a request to the Grid is submitted. It generates a proxy valid for a certain period of time. At the proxy expiration, a new proxy must be created before the Grid services can be used again.

The following sections provide details on the prerequisites.

4.1. OBTAINING A CERTIFICATE

The first requirement the user must fulfil is to be in possession of a valid X.509 certificate issued by a recognized Certification Authority (CA). The role of a CA is to guarantee that a user is who he claims to be and is entitled to own his/her certificate. It is up to the user to discover which CA he/she should contact. In general CAs are organized geographically and by research institute. Each CA has its own procedure to release certificates.

The following URL maintains an updated list of recognized CAs, as well as detailed information on how to request and install certificates of a particular CA:

http://lcg-registrar.cern.ch/pki_certificates.html

Usually, obtaining a certificate involves creating a request with the grid-cert-request command, which will generate the following files:

userkey.pem	*contains the private key associated with the certificate. (**This should be set with permissions so that only the owner can read it**) (i.e. chmod 400 userkey.pem).*
userreq.pem	*contains the request for the user certificate.*
usercert.pem	*should be replaced by the actual certificate when sent by the CA. (This should be readable by everyone) (i.e. chmod 444 usercert.pem).*

Then the userreq.pem file is sent (usually by e-mail using a particular format) to the desired CA, which will, after approval, return the new certificate also by mail.

An important property of a certificate is the ***subject****, a string containing information about the user. A typical example is:*

/O=Grid/O=CERN/OU=cern.ch/CN=John Doe

To be used in the LCG-2 Grid, the certificate must be in PEM format. If the certificate is in PKCS12 format (extension .p12), then on a machine with the openssl package installed it can be converted to PEM (extension .pem) using the pkcs12 command, in this way:

```
$ openssl pkcs12 -nocerts -in my_cert.p12 -out userkey.pem
$ openssl pkcs12 -clcerts -nokeys -in my_cert.p12 -out usercert.pem
```

where:

my_cert.p12	is the path for the input PKCS12 format file.
userkey.pem	is the path to the output private key file.
usercert.pem	is the path to the output PEM certificate file.

The first command creates only the private key (due to the -nocerts option), and the second one creates the certificate (-nokeys option). The -clcerts option instructs that only client certificates, and not CA certificates, must be created.

The grid-change-pass-phrase -file <private_key_file> command changes the passphrase that protects the private key. This command will work even if the original key is not password protected. If the -file argument is not given, the default location of the file containing the private key is assumed.

4.2. REGISTERING WITH LCG-2

Before a user can use the LCG-2 service, registration of some personal data with the LCG registration server (hosted at CERN) plus some additional steps are required. For detailed information please visit the following URL:

http://lcg-registrar.cern.ch/

To actually register oneself to the LCG-2 service, it is necessary to use a WWW browser with the user certificate for the request to be properly authenticated.

Browsers (including Internet Explorer and Netscape) use a certificate format different than the one used by the LCG-2 grid software. Browsers require a format called PKCS12 whereas grid software uses PEM format. If the certificate was issued to a user in PEM format, it has to be converted to PKCS12. The following command can be used to perform that conversion:

```
openssl pkcs12 -export -inkey userkey.pem -in usercert.pem \
-out my_cert.p12 -name "My certificate"
```

where:

userkey.pem	is the path to the private key file.
usercert.pem	is the path to the PEM certificate file.
my_cert.p12	is the path for the output PKCS12 format file to be created.
"My certificate"	is an optional name which can be used to select this certificate in the browser after the user has uploaded it if the user has more than one.

Once in PKCS12 format, the certificate can be loaded into the WWW browser. Instructions about how to do this are available at:

http://lcg-registrar.cern.ch/load_certificates.html

4.3. VIRTUAL ORGANIZATIONS

*A second requirement for the user is to belong to a **Virtual Organization (VO)**. A VO is an entity, which corresponds typically to a particular organization or group of people in the real world. The membership of a VO grants specific privileges to the user. For example, a user belonging to the ATLAS VO will be able to read the ATLAS files or to exploit resources reserved to the ATLAS collaboration.*

Entering the VO of an experiment usually requires being a member of the collaboration; the user must comply with the rules of the VO relevant to him/her to gain membership. Of course, it is also possible to be expelled from a VO when the user fails to comply with these rules.

It is not possible to access the LCG-2 Grid without being member of any VO. Every user is required to select his/her VO when registering with LCG-2 and the supplied information is forwarded to the VO administration and resource prividers for validation before the registration process is completed.

However, it is possible to belong to more than one VO at the same time. In that case, the user must choose, when submitting a job, what is the VO context for that specific job: it cannot exploit the advantage of being in two VOs at the same time.

A complete list of the VOs accepted by LCG-2 is available at the URL:

http://lcg-registrar.cern.ch/virtual_organization.html

4.4. SETTING UP THE USER ACCOUNT

To access the LCG-2 Grid, the user must also have an account on a LCG-2 User Interface.

To obtain such an account, a local system administrator must be contacted. The official list of LCG sites is available at the URL:

http://grid-deployment.web.cern.ch/grid-deployment/cgi-bin/index.cgi?var=gis/lcg2Status

As an alternative, the user can install the UI software on his/her machine (see the Installation and Administration Guide [R11]).

Once the account has been created, the user certificate must be installed. For that, it is necessary to create a directory named .globus under the user home directory and put there the user certificate and key files naming them usercert.pem and userkey.pem respectively, with permissions 0444 for the former, and 0400 for the latter.

4.5. CHECKING A CERTIFICATE

To verify that a certificate is not corrupted and print some information about it, the Globus command grid-cert-info can be used from the user's UI account. The openssl command can be used instead to verify the validity of a certificate with respect to the certificate of the certification authority that issued it.

Example 4.51 (Printing information on a user certificate)

With the certificate properly installed in the $HOME/.globus directory of the user's UI account, issue the command:

```
$ grid-cert-info
```

If the certificate is properly formed, the output will be something like:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 5 (0x5)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=CH, O=CERN, OU=cern.ch, CN=CERN CA
        Validity
            Not Before: Sep 11 11:37:57 2002 GMT
            Not After : Nov 30 12:00:00 2003 GMT
    Subject: O=Grid, O=CERN, OU=cern.ch, CN=John Doe
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:ab:8d:77:0f:56:d1:00:09:b1:c7:95:3e:ee:5d:
                    c0:af:8d:db:68:ed:5a:c0:17:ea:ef:b8:2f:e7:60:
                    2d:a3:55:e4:87:38:95:b3:4b:36:99:77:06:5d:b5:
                    4e:8a:ff:cd:da:e7:34:cd:7a:dd:2a:f2:39:5f:4a:
                    0a:7f:f4:44:b6:a3:ef:2c:09:ed:bd:65:56:70:e2:
                    a7:0b:c2:88:a3:6d:ba:b3:ce:42:3e:a2:2d:25:08:
                    92:b9:5b:b2:df:55:f4:c3:f5:10:af:62:7d:82:f4:
                    0c:63:0b:d6:bb:16:42:9b:46:9d:e2:fa:56:c4:f9:
                    56:c8:0b:2d:98:f6:c8:0c:db
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            Netscape Base Url:
                http://home.cern.ch/globus/ca
            Netscape Cert Type:
                SSL Client, S/MIME, Object Signing
            Netscape Comment:
                For DataGrid use only
            Netscape Revocation Url:
```

```
            http://home.cern.ch/globus/ca/bc870044.r0
        Netscape CA Policy Url:
            http://home.cern.ch/globus/ca/CPS.pdf
    Signature Algorithm: md5WithRSAEncryption
        30:a9:d7:82:ad:65:15:bc:36:52:12:66:33:95:b8:77:6f:a6:
        52:87:51:03:15:6a:2b:78:7e:f2:13:a8:66:b4:7f:ea:f6:31:
        aa:2e:6f:90:31:9a:e0:02:ab:a8:93:0e:0a:9d:db:3a:89:ff:
        d3:e6:be:41:2e:c8:bf:73:a3:ee:48:35:90:1f:be:9a:3a:b5:
        45:9d:58:f2:45:52:ed:69:59:84:66:0a:8f:22:26:79:c4:ad:
        ad:72:69:7f:57:dd:dd:de:84:ff:8b:75:25:ba:82:f1:6c:62:
        d9:d8:49:33:7b:a9:fb:9c:1e:67:d9:3c:51:53:fb:83:9b:21:
        c6:c5
```

The grid-cert-info command takes many options. Use the -help for a full list. For example, the -subject option returns the certificate subject:

```
$ grid-cert-info -subject
/O=Grid/O=CERN/OU=cern.ch/CN=John Doe
```

Example 4.52 (Verifying a user certificate)

Just issue from the UI the command:

```
$ openssl verify -CApath /etc/grid-security/certificates  /.globus/usercert.pem
```

and if the certificate is valid, the output will be:

```
/home/doe/.globus/usercert.pem: OK
```

If the certificate of the CA which issued the user certificate is not found in -CApath, an error message like this will appear:

```
usercert.pem: /O=Grid/O=CERN/OU=cern.ch/CN=John Doe
error 20 at 0 depth lookup:unable to get local issuer certificate
```

4.6. PROXY CERTIFICATES

*At this point, the user is able to generate a **proxy certificate**. A proxy certificate is a delegated user credential that authenticates the user in every secure interaction, and has a limited lifetime: in fact, it prevents having to use one's own certificate, which could compromise its safety.*

The command to create a proxy certificate is grid-proxy-init, which prompts for the user pass phrase, as in the next example.

Example 4.61 (Creating a proxy certificate)

To create a proxy certificate, issue the command:

```
$ grid-proxy-init
```

If the command is successful, the output will be like

```
Your identity: /O=Grid/O=CERN/OU=cern.ch/CN=John Doe
Enter GRID pass phrase for this identity:
Creating proxy ..................................... Done
Your proxy is valid until: Tue Jun 24 23:48:44 2003
```

and the proxy certificate will be written in /tmp/x509up_u<uid>, where <uid> is the Unix UID of the user, unless the environment variable X509_USER_PROXY is defined (e.g. X509_USER_PROXY=$HOME/.globus/proxy), in which case a proxy with that file name will be created, if possible.

If the user gives a wrong pass phrase, the output will be

```
ERROR: Couldn't read user key. This is likely caused by either giving the wrong passphrase or bad
file permissions key file location: /home/doe/.globus/userkey.pem Use -debug for further information.
```

If the proxy certificate file cannot be created, the output will be

```
ERROR: The proxy credential could not be written to the output file.
Use -debug for further information.
```

If the user certificate files are missing, or the permissions of userkey.pem are not correct, the output is

```
ERROR: Couldn't find valid credentials to generate a proxy.
Use -debug for further information.
```

By default, the proxy has a lifetime of 12 hours. To specify a different lifetime, the -valid H:M option can be used (the proxy is valid for H hours and M minutes - -default is 12:00). The old option -hours is deprecated. When a proxy certificate has expired, it becomes useless and a new one has to be created with grid-proxy-init. Longer lifetimes imply bigger security risks, though. Use the option -help for a full listing of options.

It is also possible to print information about an existing proxy certificate, or to destroy it before its expiration, as in the following examples.

Example 4.62 (Printing information on a proxy certificate)

To print information about a proxy certificate, for example, the subject or the time left before expiration, give the command:

```
$ grid-proxy-info
```

The output, if a valid proxy exists, will be similar to

```
subject : /O=Grid/O=CERN/OU=cern.ch/CN=John Doe/CN=proxy
issuer : /O=Grid/O=CERN/OU=cern.ch/CN=John Doe
```

```
type : full
strength : 512 bits
path : /tmp/x509up_u7026
timeleft : 11:59:56
```

If a proxy certificate does not exist, the output is:

```
ERROR: Couldn't find a valid proxy.
Use -debug for further information.
```

Example 4.63 (Destroying a proxy certificate)

To destroy an existing proxy certificate before its expiration, it is enough to do

```
$ grid-proxy-destroy
```

If no proxy certificate exists, the result will be

```
ERROR: Proxy file doesn't exist or has bad permissions
Use -debug for further information.
```

Known limitations: *A person with administrator privileges on a machine can steal proxies and run jobs on the Grid.*

4.6.1. Virtual Organization Membership Service

*The **Virtual Organization Membership Service (VOMS)** is a new service that will be used to manage authorization information in VO scope. This service is still not used in LCG-2, but the reader may find references to it in some of the commands manpages, or in the literature, and therefore it is considered necessary to make a brief description of it in this manual.*

The VOMS system will be used to include VO membership and any related authorization information in a user's proxy certificate. These proxies will be said to have VOMS extensions. The user will utilize the edg-voms-proxy-init command instead of the previously described grid-proxy-init, and a VOMS server will be contacted to check the user's certificate and create a proxy certificate with VOMS information included. By using that certificate, the VO of a user will be present in every action that he/she performs. Therefore, the user will not have to specify it using a --vo option.

NOTE: *In the current release, and while VOMS is still not used, a user can specify any VO using the --vo option when submitting a job (see Chapter 5), even if he/she does not belong to that VO, and the submission may be accepted. This does not mean, however, that the user credentials are not checked before the job is allowed to be run. The specified VO is used in this case for information and configuration purposes only, but the personal certificate of the user (through his/her proxy) is checked for the authorization, and the job is aborted if the user's real VO is not supported in the destination CE.*

4.7. ADVANCED PROXY MANAGEMENT

The proxy certificates created as described in the previous section have an inconvenient: if the job does not finish before the proxy expires, it is aborted. This is clearly a problem if, for example, the user must submit a number of jobs that take a lot of time to finish: he should create a proxy certificate with a very long lifetime, fact that would increase the security risks.

To overcome this limit, a proxy credential repository system is used, which allows the user to create and store a long-term proxy certificate on a dedicated server (Proxy Server). The WMS will then be able to use this long-term proxy to periodically renew the proxy for a submitted job before it expires and until the job ends (or the long-term proxy expires).

To see if an LCG-2 site has a Proxy Server, and what its hostname is, please refer to:

http://grid-deployment.web.cern.ch/grid-deployment/cgi-bin/index.cgi?var=gis/lcg2Status

The time necessary for the proxy renewal to take place depends on the value of the GRIDMANAGER_MINIMUM_PROXY_TIME parameter, whose current value is 600 seconds (10 minutes). As the renewal process starts some time before the initial proxy expires, ***it is necessary to generate an initial proxy long enough****, or the renewal may be triggered a bit too late, after the job has failed with the following error:*

> *Status Reason: Got a job held event, reason: Globus error 131: the user proxy expired (job is still running)*

The minimum recommended time for the initial proxy is 30 minutes*, and the edg-job-* commands will not even be accepted if the lifetime of the proxy credentials in the User Interface is lower than 10 minutes. An error message like the following will be produced:*

> ***** Error: UI_PROXY_DURATION **** Proxy certificate will expire within less then 00:10 hours.*

The advanced proxy management offered by the UI of LCG-2 through the renewal feature is available via the myproxy command suite. The user must know the host name of a Proxy Server (often referred to as ***MyProxy server****). The Proxy Server node is site and VO dependent and is usually defined in the UI configuration file stored at $EDG_WL_LOCATION/etc/VOname/edg_wl_ui.conf.*

Example 4.71 (Creating a long-term proxy and storing in a Proxy Server)

To create and store a long-term proxy certificate, the user must do, for example:

> *$myproxy-init -s <host_name> -d -n*

where <host_name> is the hostname of the machine where a Proxy Server runs, the -d option instructs the server to use the subject of the certificate as the default username, and the -n option avoids the use of a passphrase to access to the long-term proxy, so that the WMS can perform the renewals automatically.

The output will be similar to:

> *Your identity: /O=Grid/O=CERN/OU=cern.ch/CN=John Doe*
> *Enter GRID pass phrase for this identity:*

```
Creating proxy ..................................... Done
Your proxy is valid until: Thu Jul 17 18:57:04 2003
A proxy valid for 168 hours (7.0 days) for user /O=Grid/O=CERN/OU=cern.ch/CN=John Doe
now exists on lxshare0207.cern.ch.
```

By default, the long-term proxy lasts for one week and the proxy certificates created from it last 12 hours. These lifetimes can be changed using the -c and the -t option, respectively.

If the -s <host_name> option is missing, the command will try to use the $MYPROXY_SERVER environment variable to determine the Proxy Server.

***ATTENTION!** If the hostname of the Proxy Server is wrong, or the service is unavailable, the output will be similar to*

```
Your identity: /O=Grid/O=CERN/OU=cern.ch/CN=John Doe
Enter GRID pass phrase for this identity:
Creating proxy ..................................... Done
Your proxy is valid until: Wed Sep 17 12:10:22 2003
Unable to connect to adc0014.cern.ch:7512
```

where only the last line reveals that an error occurred.

Example 4.72 (Retrieving information about a long-term proxy)

To get information about a long-term proxy stored in a Proxy Server, the following command may be used:

```
$ myproxy-info -s <host_name> -d
```

where the -s and -d options have the same meaning as in the previous example.

The output is similar to

```
username: /O=Grid/O=CERN/OU=cern.ch/CN=John Doe
owner: /O=Grid/O=CERN/OU=cern.ch/CN=John Doe
timeleft: 167:59:48 (7.0 days)
```

***ATTENTION!** An important thing to remember is that the user must have a valid proxy certificate on the UI, created with grid-proxy-init, to successfully interact with his long-term certificate on the Proxy server.*

Example 4.73 (Deleting a long-term proxy)

Deleting a stored long-term proxy is achieved by doing:

```
$ myproxy-destroy -s <host_name> -d
```

And the output is:

Default MyProxy credential for user /O=Grid/O=CERN/OU=cern.ch/CN=John Doe was successfully removed.

Also in this case, a valid proxy certificate must exist for the user on the UI.

5. JOB MANAGEMENT

*In the LCG-2 Grid, a user can submit and cancel jobs, query their status, and retrieve their output. These tasks go under the name of **job management**. The LCG-2 offers two different User Interfaces to accomplish these tasks. One is the Command Line Interface and the other is the Graphical User Interface (GUI).*

5.1. THE COMMAND LINE INTERFACE

*In this section, all commands available for the user to manage jobs are described. The language used to describe a job, called **Job Description Language (JDL)**, is also explained.*

For a more detailed information on all these topics, and on the different commands, please refer to [R6].

5.1.1. Job Submission

To submit a job to the LCG-2 Grid, the user must have a valid proxy certificate in the User Interface machine (as described in 4) and use the following command:

```
$ edg-job-submit <jdl_file>
```

where <jdl_file> is a file containing the job description, usually with extension .jdl.

Example 5.1.11 (Submitting a simple job)

Create a file test.jdl with these contents:

```
Executable = "/bin/hostname";
StdOutput = "std.out";
StdError = "std.err";
OutputSandbox = {"std.out","std.err"};
```

It describes a simple job consisting of executing /bin/hostname. Standard output and error are directed to the files std.out and std.err respectively, which are then transferred back to the User Interface after the job is finished, as they are in the output sandbox. The job is submitted by issuing:

```
$ edg-job-submit test.jdl
```

If the submission is successful, the output is similar to:

```
============================ edg-job-submit Success ==================================
The job has been successfully submitted to the Network Server.
Use edg-job-status command to check job current status. Your job identifier
(edg_jobId) is:
```

```
- https://lxshare0234.cern.ch:9000/rIBubkFFKhnSQ6CjiLUY8Q
==========================================================================
```

In case of failure, an error message will be displayed instead, and an exit status different from zero returned.

*The command returns to the user the job identifier (**jobId**), which defines uniquely the job and can be used to perform further operations on the job, like interrogating the system about its status, or cancelling it. The format of the jobId is:*

```
https://Lbserver_address[:port]/unique_string
```

where unique_string is guaranteed to be unique and Lbserver_address is the address of the Logging and Bookkeeping server for the job, and usually (but not necessarily) is also the Resource Broker.

***Note**: the jobId does NOT identify a web page.*

If the command returns the following error:

```
**** Error: API_NATIVE_ERROR ****
Error while calling the "NSClient::multi" native api
AuthenticationException: Failed to establish security context...

*** Error: UI_NO_NS_CONTACT ****
Unable to contact any Network Server
```

It means that there are authentication problems between the UI and the network server (check your proxy or have the site administrator check the certificate of the server).

Many options are available to edg-job-submit.

If the user's proxy does not have VOMS extensions[2], he/she can specify his virtual organization with the -vo <vo_name> option; otherwise the default VO specified in the standard configuration file ($EDG_WL_LOCATION/etc/edg_wl_ui_cmd_var.conf) is used.

***Note:** The above mentioned configuration file can leave the default VO with a value of "unspecified". In that case, if the –vo option is not used with edg-job-submit, the command will return the following error:*

```
**** Error: UI_NO_VO_CONF_INFO ****
Unable to find configuration information for VO "unspecified"

*** Error: UI_NO_VOMS ****
Unable to determine a valid user's VO
```

where the absence of VOMS extensions in the user's proxy is also shown.

The useful -o <file_path> option allows users to specify a file to which the jobId of the submitted job will be appended. This file can be given to other job management commands to perform operations on more than one job with a single command.

[2] and currently this must be the case

The -r <CE_Id> option is used to directly send a job to a particular CE. The drawback is that the BrokerInfo functionality (see Section 5.1.8) will not be carried out. That is, the BrokerInfo file, which provides information about the evolution of the job, will not be created.

The CE is identified by <CE_Id>, which is a string with the following format:

<full_hostname>:<port_number>/jobmanager-<service>-<queue_name>

where <full_hostname> and <port> are the hostname of the machine and the port where the Globus Gatekeeper is running (the Grid Gate), <queue_name> is the name of one of the queue of jobs available in that CE, and the <service> could refer to the LRMS, such as lsf, pbs, condor, but can also be a different string as it is freely set by the site administrator when the queue is set-up.

An example of CE Id is:

adc0015.cern.ch:2119/jobmanager-lcgpbs-infinite

Similarly, the -i <file_path> allows users to specify a list of CEs from where the user will have to choose a target CE interactively.

Lastly the --nomsgi option makes the command display neither messages nor errors on the standard output. Only the jobId assigned to the job is printed to the user if the command was successful. Otherwise the location of the generated log file containing error messages is printed on the standard output. This option has been provided to make easier use of the edg-job-submit command inside scripts as an alternative to the -o option.

Example 5.1.12 (Listing computing elements that match a job description)

It is possible to see which CEs are eligible to run a job specified by a given JDL file using the command edg-job-list-match:

```
$ edg-job-list-match test.jdl
Connecting to host lxshare0380.cern.ch, port 7772
Selected Virtual Organisation name (from UI conf file): dteam

**********************************************************************
                    COMPUTING ELEMENT IDs LIST
The following CE(s) matching your job requirements have been found:

                    *CEId*
adc0015.cern.ch:2119/jobmanager-lcgpbs-infinite
adc0015.cern.ch:2119/jobmanager-lcgpbs-long
adc0015.cern.ch:2119/jobmanager-lcgpbs-short
**********************************************************************
```

The -o <file path> option can be used to store the CE list on a file, which can later be used with the -i <file path> option of edg-job-submit.

5.1.2. Job Description Language

In LCG-2, job description files (.jdl files) are used to describe jobs for execution on Grid. These files are written using a Job Description Language (JDL). The JDL adopted within the LCG-2 Grid is the ***Classified Advertisement (ClassAd) language****[R12] defined by the* ***Condor Project*** *[R13], which deals with the management of distributed computing environments, and whose central construct is the* ***ClassAd****, a record-like structure composed of a finite number of distinct attribute names mapped to expressions. A ClassAd is a highly flexible and extensible data model that can be used to represent arbitrary services and constraints on their allocation. The JDL is used in LCG-2 to specify the desired job characteristics and constraints, which are used in by match-making process to select the resources that the job will use.*

The fundamentals of the JDL are given in this section. A detailed description of the JDL syntax is out of the scope of this guide, and can be found in [R14] and [R15].

The JDL syntax consists on staments like:

```
attribute = value;
```

ATTENTION!!! *The JDL is sensitive to blank characters and tabs. No blank characters or tabs should follow the semicolon at the end of a line.*

In a job description file, some attributes are mandatory, while some others are optional. Essentially, one must at least specify the name of the executable, the files where to write the standard output and the standard error of the job (they can even be the same file). For example:

```
Executable = "test.sh";
StdOutput = "std.out";
StdError = "std.err";
```

If needed, arguments to the executable can be passed:

```
Arguments = "hello 10";
```

If the argument list contains quoted strings, the quotes must be escaped with a backslash (e.g. Arguments = " \"hello\" 10"). For special characters, such as &, the shell on the WN will itself expect the escaped form: \&, and therefore both the slash and the ampersand will have to be escaped inside the JDL file, resulting in: \\\&. In general, special characters such as &, |, >, < are only allowed if specified inside a quoted string or preceded by triple \. The character " ' " cannot be specified in the JDL.

For the standard input, an input file can be similarly specified (though this is not required):

```
StdInput = "std.in";
```

Then, the files to be transferred between the UI and the WN before (Input Sandbox) and after (Output Sandbox) the job execution can be specified:

```
InputSandbox = {"test.sh","std.in"};
OutputSandbox = {"std.out","std.err"};
```

Wildcards are allowed only in the InputSandbox attribute. The list of files in the Input Sandbox is specified relatively to the current working directory. Absolute paths cannot be specified in the OutputSandbox attribute.

Neither the InputSandbox nor the OutputSandbox lists can contain two files with the same name (even if in different paths) as when transferred they would overwrite each other.

***Note:** The executable flag is not preserved for the files included in the Input Sandbox when transferred to the WN. Therefore, for any file needing execution permissions a chmod +x operation should be performed by the initial script specified as the Executable in the JDL file (the chmod +x operation is done automatically for this script).*

The environment of the job can be modified using the Environment attribute. For example:

```
Environment = {"CMS_PATH=$HOME/cms", "CMS_DB=$CMS_PATH/cmdb"};
```

If the job requires some files stored in an LCG storage element, the InputData attribute can be used to make the resource broker select a CE as close as possible to the files. The OutputSE attribute, similarly, specifies the SE where the user wants to store the generated output data. This is used by the RB to find a CE that is close to the given SE. Finally, the OutputData attribute can be used to automatically have any output data files copied and registered in the Grid.

Example 5.1.21 (Specifying input data in a job)

If the user job needs to read two files (identified by a logical file name or by their GUID), the job description file may contain a line like the following:

```
InputData = {"lfn:doe/prod/kin_1", "guid:136b48a64-4a3d-87ud-3bk5-8gnn46m49f3"};
```
[3]

In addition, if the InputData attribute is used, the protocols the application is able to use to read the files must be declared. The only supported protocols are gsiftp (the GSI version of ftp) and rfio.

```
DataAccessProtocol = {"rfio", "gsiftp"};
```

The meaning of these two protocols is the following:

- ***gsiftp**: this is the Grid-enabled ftp protocol. It means that the application will access the file via ftp, for instance copying the file first on a locally accessible storage.*
- ***rfio**: allows the user to read and write files remotely, but only within a local area network (not between sites), as it is not included in the GSI yet. It is just a read-write remote protocol.*

The inclusion of these attributes will cause the Resource Broker to look for replicas of the specified files, in order to find a CE which can access them in a close SE. If the are no accessible replicas (in a close SE), the submission will fail.

However, if the user knows that there are replicas of the required files in a distant SE, he/she can copy them manually to a close SE beforehand, so the submission works. It is planned that in future releases this cam be done automatically by the Grid. Moreover, the user can also leave these attributes out of the jdl file, and still access the files from the job, using gsiftp (rfio will not work if files are located in different local area networks). This is, though, against the philosophy of the Grid, since a CE should not access distant files, increasing the network traffic, but rather use closer copies.

[3] For details on file names conventions refer to 3.1.5

Detailed information of how the job can access the grid files is given in Chapter 6.

The job will be sent to the CE with the best rank (which is a user-definable measurement of the CE goodness), between all the CEs satisfying all the job requirements and having the maximum number of file replicas on a SE close to them.

Example 5.1.22 (Specifying a Storage Element)

The user can ask the job to run close a specific storage element, in order to store there the output data, using the attribute OutputSE. For example:

```
OutputSE = "lxshare0291.cern.ch";
```

The Resource Broker will not abort the job if there is no CE close to the OutputSE specified by the user. The RB will try to find resources close to such SE but if the CE cannot be found the job will run somewhere else.

Example 5.1.23 (Automatic upload and registration of output files)

The OutputData attribute allows the user for the automatic upload and registration in LCG-2 of files produced by the job on the WN. Several output files can be specified. For each of this files, three attributes can be set.

The OutputFile attribute is mandatory and specifies the name of the generated file to be uploaded to the Grid. The StorageElement is an optional string indicating the SE where the file should be stored. If unspecified, the WMS automatically choses a SE close to the CE. Finally, the LogicalFileName attribute (also optional) represents a LFN the user wants to be associated to the output file in LCG-2.

The following code shows an example OutputData attribute:

```
OutputData = {
        [
        OutputFile="my_file_1.out";
        LogicalFileName="lfn:my_test_result"
        StorageElement="lxshare0291.cern.ch"
        ],
        [
        OutputFile="my_file_2.out"
        LogicalFileName="my_debugging"
        ]
};
```

To express any kind of requirement on the resources where the job can run, there is the Requirements attribute. Its value is a Boolean expression that must evaluate to true for a job to run on that specific CE. For that purpose all the GLUE attributes of the IS can be used. For a list of GLUE attributes, see Appendix A.

Example 5.1.24 (Specifying requirements on the CE)

Let us suppose that the user wants to run on a CE using PBS as the LRMS, and whose WNs have at least two CPUs. He will write then in the job description file:

```
Requirements = other.GlueCEInfoLRMSType == "PBS" && other.GlueCEInfoTotalCPUs > 1;
```

The WMS can be also asked to send a job to a particular CE with the following expression:

```
Requirements = other.GlueCEUniqueID == "lxshare0286.cern.ch:2119/jobmanager-pbs-short";
```

If the job must run on a CE where a particular experiment software is installed and this information is published by the CE, something like the following must be written:

```
Requirements = Member("CMSIM-133", other.GlueHostApplicationSoftwareRunTimeEnvironment);
```

***Note**: The Member operator is used to test if its first argument (a scalar value) is a member of its second argument (a list). In this example, the GlueHostApplicationSoftwareRunTimeEnvironment attribute is a list.*

As a general rule, requirements on attributes of a CE are written prefixing "other." to the attribute name in the Information System schema.

Example 5.1.25 (Specifying requirements using wildcards)

It is also possible to use regular expressions when expressing a requirement. Let us suppose for example that the user wants all this jobs to run on CEs in the domain cern.ch. This can be achieved putting in the JDL file the following expression:

```
Requirements = RegExp("cern.ch", other.GlueCEUniqueId);
```

The opposite can be required just by using

```
Requirements = (!RegExp("cern.ch", other.GlueCEUniqueId));
```

Example 5.1.26 (Specifying requirements on the close SE)

*The previous requirements affected always two entities: the job and the CE. In order to specify requirements involving three entities (i.e., the job, the CE and the SE), the RB uses a special match-making mechanism, called **gangmatching**. This is supported by some JDL functions: anyMatch, whichMatch, allMatch. A typical example of this functionality follows. For information on the gangmatching, please refer to [R15].*

To ensure that the job runs on a CE with, for example, at least 200 MB of free disk space in a close SE, the following JDL expression can be used[4]*:*

[4]The function used to calculate the available space in a SE can be inaccurate if the SE uses NFS mounted filesystems. Also, the measurement is not useful for SE using MSS (such as tape systems), as the available space returned is infinite (or 1000000000000), since new tapes can always be added

Requirements = anyMatch(other.storage.CloseSEs, target.GlueSAStateAvailableSpace > 204800);

The VirtualOrganisation attribute represents another way to specify the VO of the user, as for example in:

VirtualOrganisation = "cms";

A common error is to write VirtualOrganization: it will not work.

This value is anyway superseded by the -vo option of edg-job-submit.

The JDL attribute called RetryCount can be used to specify how many times the WMS must try to resubmit a job if it fails due to some LCG component (that is, not the job itself). The default value (if any) is defined in the file $EDG_WL_LOCATION/etc/edg_wl_ui_cmd_var.conf.

The MyProxyServer attribute indicates the Proxy Server containing the user's long-term proxy that the WMS must use to renew the proxy certificate when it is about to expire.

The choice of the CE where to execute the job, among all the ones satisfying the requirements, is based on the rank of the CE; namely, a quantity expressed as a floating-point number. The CE with the highest rank is the one selected.

The user can define the rank with the Rank attribute as a function of the CE attributes, like in the following (which is also the default definition):

Rank = other.GlueCEStateFreeCPUs;

5.1.3. Job Operations

After a job is submitted, it is possible to see its status and its history, and to retrieve logging information about it. Once the job is finished the job's output can be retrieved, although it is also possible to cancel it previously. The following examples explain how.

Example 5.1.31 (Retrieving the status of a job)

Given a submitted job whose job identifier is <jobId>, the command is:

$ edg-job-status <jobId>

And an example of a possible output is

```
**********************************************************
BOOKKEEPING INFORMATION:

Printing status info for the Job:
https://lxshare0234.cern.ch:9000/X-ehTxfdlXxSoIdVLS0L0w

Current Status: Ready
```

```
Status Reason: unavailable
Destination: lxshare0277.cern.ch:2119/jobmanager-pbs-infinite
reached on: Fri Aug 1 12:21:35 2003
*************************************************************
```

where the current status of the job is showed, along with the time when that status was reached, and the reason for being in that state (which may be especially helpful for the ***ABORTED*** *state). The possible states in which a job can be found were introduced in Section 3.2.1, and are summarised in Appendix C. Finally, the destination field contains the ID of the CE where the job has been submited.*

Much more information is provided if the verbosity level is increased by using -v1 or -v2 with the command. See [R6] for detailed information on each of the fields that are returned then.

Many job identifiers can be given as arguments of the edg-job-status command, i.e.:

```
edg-job-status <jobId1> ... <jobIdN>
```

The option -i <file path> can be used to specify a file with a list of job identifiers (saved previously with the -o option of edg-job-submit). In this case, the command asks the user interactively the status of which job(s) should be printed. Subsets of jobs can be selected (e.g. 1-2,4).

```
$ edg-job-status -i jobs.list
_____________________________________________
1 : https://lxshare0234.cern.ch:9000/UPBqN2s2ycxt1TnuU3kzEw
2 : https://lxshare0234.cern.ch:9000/8S6IwPW33AhyxhkSv8Nt9A
3 : https://lxshare0234.cern.ch:9000/E9R0Yl4J7qgsq7FYTnhmsA
4 : https://lxshare0234.cern.ch:9000/Tt80pBn17AFPJyUSN9Qb7Q
a : all
q : quit
_____________________________________________

Choose one or more edg_jobId(s) in the list - [1-4]all:
```

If the --all option is used insted, the status of all the jobs owned by the user submitting the command is retrieved.

NOTE: *for the --all option to work, it is necessary that an index by owner is created in the LB server; otherwise, the command will fail, since it will not be possible for the LB server to identify the user's jobs. Such index can only be created by the LB server administrator, as explained in section 5.2.2 of [R6].*

With the option -o <file path> the command output can be written to a file.

Example 5.1.32 (Cancelling a job)

A job can be can be cancelled before it ends using the command edg-job-cancel.

This command requires as arguments one or more job identifiers. For example:

```
$ edg-job-cancel https://lxshare0234.cern.ch:9000/dAE162is6EStca0VqhVkog \
```

https://lxshare0234.cern.ch:9000/C6n5Hq1ex9-wF2t05qe8mA

Are you sure you want to remove specified job(s)? [y/n]n :y
========================= edg-job-cancel Success==================
The cancellation request has been successfully submitted for the following job(s)
- https://lxshare0234.cern.ch:9000/dAE162is6EStca0VqhVkog
- https://lxshare0234.cern.ch:9000/C6n5Hq1ex9-wF2t05qe8mA

All the command options work exactly as in edg-job-status.

Example 5.1.33 (Retrieving the output of a job)

After the job has finished (it reaches the ***Done*** *status), its output can be copied to the UI with the command edg-job-get-output, which takes a list of jobs as argument. For example:*

$ edg-job-get-output https://lxshare0234.cern.ch:9000/snPegp1YMJcnS22yF5pFlg
Retrieving files from host lxshare0234.cern.ch

****
JOB GET OUTPUT OUTCOME

Output sandbox files for the job:
- https://lxshare0234.cern.ch:9000/snPegp1YMJcnS22yF5pFlg
have been successfully retrieved and stored in the directory:
/tmp/jobOutput/snPegp1YMJcnS22yF5pFlg

****

By default, the output is stored under /tmp, but it is possible to specify in which directory to save the output using the --dir <path_name> option.

All command options work exactly as in edg-job-status.

Example 5.1.34 (Retrieving logging information about submitted jobs)

The edg-job-get-logging-info command queries the LB persistent database for logging information about jobs previously submitted using edg-job-submit. The job's logging information is stored permanently by the LB service and can be retrieved also after the job has terminated its life-cycle. This is especially useful in the analysis of job failures.

The argument of this command is a list of one or more job identifiers. The -i and -o options work as in the previous commands. As an example consider:

edg-job-get-logging-info -v 0 -o logfile.txt https://lxshare0310.cern.ch:9000/C_CBUJKqc6Zqd4clQaCUTQ

================ edg-job-get-logging-info Success ==================

Logging Information has been found and stored in the file:
/afs/cern.ch/user/d/delgadop/pruebas/logfile.txt
==

where the --v option sets the detail level of information about the job displayed to the user (possible values are 0,1 and 2).

The output (stored in the file logfile.txt) will be:

****
LOGGING INFORMATION:

Printing info for the Job: https://lxshare0310.cern.ch:9000/C_CBUJKqc6Zqd4clQaCUTQ

- - -
Event: RegJob
- source = UserInterface
- timestamp = Fri Feb 20 10:30:16 2004
- - -
Event: Transfer
- destination = NetworkServer
- result = START
- source = UserInterface
- timestamp = Fri Feb 20 10:30:16 2004
- - -
Event: Transfer
- destination = NetworkServer
- result = OK
- source = UserInterface
- timestamp = Fri Feb 20 10:30:19 2004
- - -
Event: Accepted
- source = NetworkServer
- timestamp = Fri Feb 20 10:29:17 2004
- - -
Event: EnQueued
- result = OK
- source = NetworkServer
- timestamp = Fri Feb 20 10:29:18 2004
. . .

5.1.4. Checkpointable Jobs

Checkpointable jobs *are specified setting the JDL JobType attribute to Checkpointable. When a checkpointable job is submitted the user can specify the number (or list) of steps in which the job can be logically decomposed and the step to be considered as the initial one. This can be done setting respectively the JDL attributes JobSteps and CurrentStep. The CurrentStep attribute is a mandatory attribute and if not provided by the user, it is set automatically to 0 by the UI.*

The --chkpt option allows the submission of a checkpointable job specifying as input a checkpoint state generated by a previously submitted job. This option makes the submitted job start running from the checkpoint state given in input and not from the very beginning.

The initial checkpoint states to be used with this option can be retrieved by means of the edg-job-get-chkpt command. Checkpoint states are retrieved from the LB server and are saved locally into a file in JDL format, as described in [R16].

The --cs <state_number> option allows the user to select the checkpoint state she/he wants to be retrieved. For instance, using --cs N causes the command to retrieve the last but N job checkpoint state that was saved. Otherwise, the last saved state is retrieved.

5.1.5. Interactive Jobs

***Interactive jobs** are specified setting the JDL JobType attribute to Interactive. When an interactive job is submitted, the edg-job-submit command starts a grid console shadow process in the background that listens on a port for the job standard streams. Moreover the edg-job-submit command opens a new window where the incoming job streams are forwarded. The port on which the shadow process listens is assigned by the OS, but can be forced through the ListenerPort attribute in the JDL.*

As the command in this case opens a X window, the user should make sure the DISPLAY environment variable is correctly set, a X server is running on the local machine and, if she/he is connected to the UI node from remote machine (e.g. with ssh), enable secure X11 tunneling. If this is not possible, the user can specify the --nogui option that makes the command provide a simple standard non-graphical interaction with the running job.

Another option that is reserved for interactive jobs is --nolisten: it makes the command forward the job standard streams coming from the WN to named pipes on the UI machine whose names are returned to the user together with the OS id of the listener process. This allows the user to interact with the job through her/his own tools. It is important to note that when this option is specified, the UI has no more control over the launched listener process that has hence to be killed by the user (through the returned process id) when the job is finished.

The listener process and the window are started automatically by the edg-job-submit command for interactive jobs. In the case that the interactive session with a job is lost, or if the user needs to follow the job from a different machine (not the UI), or on another port, a new interactive session can be started with the edg-job-attach command. This commands starts a listener process on the UI machine that is attached to the standard streams of a previously submitted interactive job and displays them on a dedicated window. The --port <port_number> option specifies the port on which the listener is started.

5.1.6. MPI Jobs

***NOTE:** MPI software has not been tested yet, and it is not part of the official distribution of the current LCG-2 relase. Any site installing or using it will do it only under its own responsability.*

This section gives a brief overview of how MPI jobs should work in LCG-2.

***Message Passing Interface (MPI)** applications are run in parallel in several processors. Jobs that must be run*

as MPI are specified setting the JDL JobType attribute to MPICH. When a MPI job is submitted, the presence of the NodeNumber attribute (it specifies the required number of CPUs) in the JDL is mandatory and the UI automatically requires the MPICH runtime environment installed on the CE and a number of CPUs at least equal to the required number of nodes. This is done adding the following expression:

```
(other.GlueCEInfoTotalCPUs >= NodeNumber) &&
Member(other.GlueHostApplicationSoftwareRunTimeEnvironment,"MPICH")
```

to the the JDL requirements expression.

5.1.7. Advanced Command Options

All the edg-job-* commands read some configuration files which the user can edit, if he/she is not satisfied with the default ones.

The main configuration file is located by default at $EDG_WL_LOCATION/etc/edg_wl_ui_cmd_var.conf, and sets, among other things, the default VO, the default location for job outputs and command log files and the default values of mandatory JDL attributes. It is possible to point to a different configuration file by setting the value of the environment variable $EDG_WL_UI_CONFIG_VAR to the file path, or by specifying the file in the --config <file> option of the edg-job-* commands (which takes precedence).

In addition, VO-specific configurations are defined by default in the file $EDG_WL_LOCATION/etc/<vo>/edg_wl_ui.conf, consisting essentially in the list of Network Servers, Proxy Servers and LB servers accessible to that VO. A different file can be specified using the variable $EDG_WL_UI_CONFIG_VO or the --config-vo <file> option of the edg-job-* commands.

Example 5.1.71 (Changing the default VO)

A user can change his/her default VO by performing the following steps:

a. Make a copy of the file $EDG_WL_LOCATION/etc/edg_wl_ui_cmd_var.conf, for example to $HOME/my_ui.conf.
b. Edit $HOME/my_ui.conf and change this line:

   ```
   DefaultVo = "cms";
   ```

   if, for example, he wants to set the CMS VO as default.
c. Define in the shell configuration script ($HOME/.bashrc for bash and $HOME/.cshrc for csh/tcsh) the environment variable

   ```
   setenv EDG_WL_UI_CONFIG_VAR $HOME/my_ui.conf      ((t)csh)
   export EDG_WL_UI_CONFIG_VAR=$HOME/my_ui.conf      (bash)
   ```

The --log <file> option allows the user to define the log file; the default log file is named <command_name>_<UID>_<PID>_<date_time>.log and it is found in the directory specified in the configuration file. The --noint option skips all interactive questions and prints all warning and error messages to a log file. The --help and --version options are self-explanatory.

5.1.8. The BrokerInfo

The **BrokerInfo file** is a mechanism by which the user job can access, at execution time, certain information concerning the job, for example the name of the CE, the files specified in the InputData attribute, the SEs where they can be found, etc.

The BrokerInfo file is created in the job working directory (that is, the current directory on the WN for the executable) and is named .BrokerInfo. Its syntax is, as in job description files, based on Condor ClassAds and the information contained is not easy to read; however, it is possible to get it by means of a CLI, whose description follows.

Detailed information about the BrokerInfo file, the edg-brokerinfo CLI, and its respective API can be found in [R17].

The edg-brokerinfo command has the following syntax:

edg-brokerinfo [-v] [-f <filename>] function [parameter] [parameter] ...

where function is one of the following:

- getCE: returns the name of the CE the job is running on;
- getDataAccessProtocol: returns the protocol list specified in the DataAccessProtocol JDL attribute;
- getInputData: returns the file list specified in the InputData JDL attribute;
- getSEs: returns the list of the storage elements with contain a copy of at least one file among those specified in InputData;
- getCloseSEs: returns a list of the storage elements close to the CE;
- getSEMountPoint <SE>: returns the access point for the specified <SE>, if it is the list of clode SEs of the WN.
- getSEFreeSpace <SE>: returns the free space on <SE>;
- getLFN2SFN <LFN>: returns the storage file name of the file specified by <LFN>, where <LFN> is a logical file name of a GUID specified in the InputData attribute;
- getSEProtocols <SE>: returns the list of the protocols available to transfer data in the storage element <SE>;
- getSEPort <SE> <Protocol>: returns the port number used by <SE> for the data transfer protocol <Protocol>;
- getVirtualOrganization: returns the name of the VO specified in the VirtualOrganization JDL attribute.
- getAccessCost: not supported at present.

The -v option produced a more verbose output, and the -f <filename> option tells the command to parse the BrokerInfo file specified by <filename>. If the -f option is not used, the command tries to parse the file $EDG_WL_RB_BROKERINFO.

There are basically two ways for parsing elements from a BrokerInfo file.

The first one is directly from the job, and therefore from the WN where the job is running. In this case, the $EDG_WL_RB_BROKERINFO variable is defined as the location of the .BrokerInfo file, in the working directory of the job, and the command will work without problems. This can be accomplished for instance by including a line like the following in a submitted shell script:

```
/opt/edg/bin/edg-brokerinfo getCE
```

where the edg-brokerinfo command is called with any desired function as its argument.

If, on the contrary, edg-brokerinfo is invoked from the UI, the $EDG_WL_RB_BROKERINFO variable will be usually undefined, and an error will occur. The solution to this is to include an instruction to generate the .BrokerInfo file as ouput of the submitted job, and retrieve it with the rest of generated output, when the job finishes. This can be done for instance with:

```
# !/bin/sh cat $EDG_WL_RB_BROKERINFO
```

in a submitted shell script.

Then, the file can be accessed locally with the -f option commented above.

5.2. THE GRAPHICAL USER INTERFACE

The EDG WMS GUI is a Java Graphical User Interface composed of three different applications: the JDL Editor, the Job Monitor and the Job Submitter. The 3 GUI components are integrated although they can be used as standalone applications so that the JDL Editor and the Job Monitor can be invoked from the Job Submitter, thus providing a comprehensive tool covering all main aspects of job management in a Grid environment: from creation of job descriptions to job submission, monitoring and control up to output retrieval.

Details on the EDG WMS GUI are not given in this guide. Please refer to [R18] for a complete description of the functionalities provided by the GUI, together with some example screenshots.

6. DATA MANAGEMENT

6.1. INTRODUCTION

6.1.1. EDG Data Management Tools

In this chapter, the EDG Data Management tools are described. These are high level tools used to upload files to the grid, replicate data and locate the best replica available. Some use cases and example usage for theses tools are listed. Besides, some lower level tools (like edg-gridftp- commands) are introduced. These low level tools should only be used in case of problems and anyway by system administrators only and not by LCG-2 Grid users. As a reference only, a brief summary on their functions will be given.*

The Data Management tools are:

edg-replica-manager (edg-rm)	*client tools*
edg-local-replica-catalog (edg-lrc)	*client tools*
edg-replica-metadata-catalog (edg-rmc)	*client tools*

For details on how to use the client tools mentioned above, please refer to [R19], [R20], [R21]. In addition, more detailed examples on Replica Manager usage can be found in [R22].

Apart from those presented above, there are two more commands in the UI: the edg-replica-location-index (edg-rli) and the edg-replica-optmization (edg-ros). These two commands will allow the user to interact with the Replica Location Index and the Replica Optimization services, when they are in work. These services are planned for the LCG architecture, but they are not in use yet (and their commands are therefore not useful at the moment). Information about these two commands can be found in [R23] and [R24].

6.1.2. File Names within LCG-2

As a reminder of what was explained in Chapter 3, the different types of names that can be used withing the LCG-2 files catalogues are summarized as follows:

A GUID, which identifies a file uniquely, is of the form:

guid:<40_bytes_unique_string>

like:

guid:38ed3f60-c402-11d7-a6b0-f53ee5a37e1d

An LFN or User Alias, which can be used to refer to a file in the place of the GUID, has this format:

lfn:<anything_you_want>

like:

lfn:importantResults/Test1240.dat

A SURL, which identifies a replica in a SE, is of the form:

sfn://<SE_hostname><SE_Accesspoint><VO_path><filename>

like:

sfn://tbed0101.cern.ch/flatfiles/SE00/dteam/generated/2004-02-26/
file3596e86f-c402-11d7-a6b0-f53ee5a37e1d

Finally, a TURL, which is a valid URI with the necessary information to access a file in a SE, has the following form:

<protocol>://<SE_hostname><SE_Accesspoint><VO_path><filename>

like:

gsiftp://tbed0101.cern.ch/flatfiles/SE00/dteam/generated/2004-02-26/
file3596e86f-c402-11d7-a6b0-f53ee5a37e1d

Failure to comply with these rules results in corrupted catalogues and malfunctioning replica management.

6.2. EDG-REPLICA-MANAGER CLIENT TOOLS

The EDG Replica Manager client tools allow users to copy files between UI, CE, WN and a SE, to register entries in the RLS and replicate files between SEs. There are different commands that are invoked using:

$ edg-rm --vo <vo_name> <cmd_name>

where the --vo <vo_name> option specifies the virtual organization of the user (this option is mandatory —without it the command will not work), and <cmd_name> is the particular command that the RM must perform. Most commands have both an extended and an abbreviated name form.

Other general edg-rm options are: --log-debug, --log-info and --log-off, which are used for enabling or disabling bug-level or info-level logging; and the --config <file> option, which is used to read the specified configuration file, instead of the default $EDG_LOCATION/etc/edg-replica-manager/edg-replica-manager.conf.

Note: In the current release, if a local file called edg-replica-manager.conf exists, the RM will use it as configuration file even if it is not specified by the user with the --config option.

In what follows some usage examples are given. For details on the options of each command, please use the --help option with edg-rm. If the name of a command is also given, then specific information about that command is presented. The user can also consult the manpages and [R19].

For clarity reasons, in the pieces of code that follow (throghout the whole chapter), the commands introduced by the user are leaded by a '$' symbol, and the answers of the shell are usually preceded by '>' (unless the difference is obvious).

6.2.1. Basic Replica Manager Commands

Example 6.2.11 (Uploading a file from the UI to the Grid)

In order to upload a file to the Grid, i.e., to transfer it from the local machine to a Storage Element where it must reside permanently, the CopyAndRegister (cr) command can be used (in a machine with a valid proxy):

```
$ edg-rm --vo dteam file:///home/antonio/file1 cr -l lfn:my_alias1
> guid:6ac491ea-684c-11d8-8f12-9c97cebf582a
```

where the only argument is the local file to be uploaded (a fully qualified URI) and the -l option indicates an LFN for it. The command returns the unique GUID for the file. If no LFN is provided, then the returned GUID will be the only way to access the file in the Grid.

If the -d <destination> option is included, then the specified SE (which must be known in advance) is used as the destination for the file. Without the -d option, a default SE is chosen automatically. A complete SURL, including the SE hostname, the path (accesspoint plus VO-specific directory) and a chosen filename, or only the SE hostname can be used as the destination. This is illustrated by the following commands:

```
$ edg-rm --vo dteam file:/home/antonio/file1 cr -l lfn:my_alias1 -d tbed0115.cern.ch
```

or

```
$ edg-rm --vo dteam file:/home/antonio/file1 cr -l lfn:my_alias1 \
-d sfn://tbed0115.cern.ch/dteam/my_file1
```

In this and other commands the -p <protocol> and -n <#streams> options can be used to specify the protocol (gsiftp being the default one) and the number of parallel streams to be used in the transfer (default is 8).

Example 6.2.12 (Retrieving information about the Grid)

If the above described -d option is to be used, then the information about the available SEs must be retrieved in advance. There are several ways to retrieve information about the resources on the Grid. Either the Information Service is queried directly (as explained in Chapter 7), or the EDG Replica Manager printInfo (pi) command is used:

```
$ edg-rm --vo=dteam printInfo
```

The previous command returns all Ces and SEs that the Replica Manager retrieves from the IS, as well as the RMC and LRC used in the specified VO. The name of all the queues is also given for every CE, along with the SEs that are close to it. Regarding the SEs, the VOs and protocols supported, and their accesspoint are provided.

A typical output is as follows:

```
VO used       : cms
default SE    : tbed0101.cern.ch
default CE    : pceis01.cern.ch
```

```
Info Service      : MDS

RMC endpoint   : http://rlscert01.cern.ch:7777/dteam/v2.2/edg-replica-metadata-catalog/services/
                 edg-replica-metadata-catalog
LRC endpoint   : http://rlscert01.cern.ch:7777/edg-replica-location/services/
                 /edg-local-replica-catalog
ROS endpoint   : no information found: No Service found edg-replica-optimization

List of CE ID's: pceis01.cern.ch:2119/jobmanager-pbs-infinite
                 pceis01.cern.ch:2119/jobmanager-pbs-long
                 pceis01.cern.ch:2119/jobmanager-pbs-medium
                 pceis01.cern.ch:2119/jobmanager-pbs-short
. . .

CE at infinite :
            name : infinite
            ID: pceis01.cern.ch:2119/jobmanager-pbs-infinite
            closeSEs : cmslcgse02.cern.ch,lcgse02.ifae.es,tbed0101.cern.ch,tbed0115.cern.ch,
                       wacdr002d.cern.ch
            VOs : alice,atlas,cms,lhcb,dteam
. . .

List of SE ID's : tbed0101.cern.ch
                  tbed0115.cern.ch
                  wacdr002d.cern.ch
                  cmslcgse02.cern.ch
                  lcgse02.ifae.es

SE at eis :
      name : eis
      host : tbed0101.cern.ch
      type : disk
      accesspoint : /flatfile/SE00
      VOs : alice,atlas,cms,dteam,lhcb
      VO directories : alice:/alice,atlas:/atlas,cms:/cms,dteam:/dteam,lhcb:/lhcb
      protocols : gsiftp,rfio
. . .
```

In order to find all SEs, their access point and the VO directories the user can filter the previous response with grep, as in the following example, where the desired information is specified with the -e option and, just to get a nicer output, unwaanted lines are eliminated by using the -v option.

```
$ edg-rm --vo=dteam pi | grep -e SE -e host -e accesspoint
-e 'VO directories' | grep -v closeSEs | grep -v "List of SE"

default SE : tbed0101.cern.ch

SE at eis :
            host : tbed0101.cern.ch
```

```
        accesspoint : /flatfile/SE00
        VO directories : alice:/alice,atlas:/atlas,cms:/cms,dteam:/dteam,lhcb:/lhcb
SE at eis :
        host : tbed0115.cern.ch
        accesspoint : /
        VO directories : alice:alice,atlas:atlas,cms:cms,dteam:dteam,lhcb:lhcb
SE at CERN-LCG2 :
        host : wacdr002d.cern.ch
        accesspoint : /castor/cern.ch/grid
        VO directories : alice:alice,atlas:atlas,cms:cms,dteam:dteam,lhcb:lhcb
SE at eis :
        host : cmslcgse02.cern.ch
        accesspoint : /data1/lcg
        VO directories : cms:cms
SE at PIC-LCG2 :
        host : lcgse02.ifae.es
        accesspoint : /castor/ifae.es/lcg
        VO directories : atlas:atlas,cms:cms,dteam:dteam,lhcb:lhcb
```

The printInfo command does not return the free space on the SE. That information can be obtained by directly querying the Information Service.

Example 6.2.13 (Replicating a file)

Once a file is stored on an SE and registered with the Replica Location Service, the file can be replicated using the replicateFile (rep) command, as in:

```
$ edg-rm --vo=dteam replicateFile guid:6ac491ea-684c-11d8-8f12-9c97cebf582a \
-d wacdr002d.cern.ch
> sfn://wacdr002d.cern.ch/castor/cern.ch/grid/dteam/generated/2004-02-26/ filea778c4f6-687d-11d8-
a111-c2fed1a6363a
```

where the file to be replicated can be specified using a LFN, GUID or even a particular SURL, and the -d option is used to specify the SE where the new replica will be stored (and, as with CopyAndRegisterFile, using either the SE hostname or a complete SURL). If this option is not set, then the an SE is chosen automatically.

For one GUID, there can be only one replica per SE. If the user tries to use the replicateFile command with a destination SE that already holds a replica, the existing SURL will be returned, and no new replica will be created.

Example 6.2.14 (Listing replicas and GUIDs)

The Replica Manager allows users to list all the replicas of a file that have been successfully registered with the Replica Location Service. For that purpose the listReplicas (lr) command is used:

```
$ edg-rm --vo=dteam lr lfn:my_alias1
> sfn://tbed0101.cern.ch/flatfile/SE00/dteam/generated/2004-02-26/ filea72eaedc-684b-11d8-8efc-fc10ad029740
```

> sfn://wacdr002d.cern.ch/castor/cern.ch/grid/dteam/generated/2004-02-26/ filea778c4f6-687d-11d8-a111-c2fed1a6363a

Again, LFN, GUID or SURL can be used to specify the file for which all replicas must be listed. The SURLs of the replicas are returned.

Reciprocally, the listGUID (lg) return the GUID associated with a specified LFN or SURL:

```
$ edg-rm --vo=dteam lg sfn://tbed0101.cern.ch/flatfile/SE00/dteam/my_file1
> guid:c06a92ee-6911-11d8-a453-d9c1af867039
```

The tools edg-local-replica-catalog and edg-replica-metadata-catalog, described later, provide more functions for catalog interaction.

Example 6.2.15 (Copying files out of the Grid)

The copyFile (cp) command can be used to copy a Grid file to a non-grid storage resource. This is useful to have a local copy of the file. The command accepts the LFN, GUI or SURL of the LCG-2 file as its first argument and a local filename or valid TURL as the second, as is shown in the following example:

```
$ edg-rm --vo dteam cp lfn:my_alias2 file:/home/antonio/file2
```

Note that although this command is designed to copy files from a SE to a non-grid resources, if the proper TURL is used, a file could be transferred from one SE to another, or from out of the Grid to a SE. **This should not be done**, since it has the same effect as using replicateFile but **skipping the file registration**, making in this way this replica invisible to Grid users.

Example 6.2.16 (Obtaining a TURL for a replica)

For any given replica (identified by its SURL) the TURL for accessing it using a particular protocol can be obtained with the getTurl (gt) command. The arguments are the SURL and the protocol to be used. The command returns the valid TURL or an error message if the specified protocol is not supported by that SE for the given replica.

```
$ edg-rm --vo dteam getTurl sfn://tbed0101.cern.ch/flatfile/SE00/dteam/generated/2004-02-26/f1 gsiftp
> gsiftp://tbed0101.cern.ch/flatfile/SE00/dteam/generated/2004-02-26/f1
```

```
$ edg-rm --vo dteam getTurl sfn://tbed0101.cern.ch/flatfile/SE00/dteam/generated/2004-02-26/f1 ftp
> The file sfn://tbed0101.cern.ch/flatfile/SE00/dteam/generated/2004-02-26/f1 is not accessible via the protocol: ftp
```

Example 6.2.17 (Deleting replicas)

Once a file is stored on a Storage Element and registered with a catalog, it can be deleted using the deleteFile (del) command. If a SURL is provided as argument, then that particular replica will be deleted. If a LFN is given

instead, then the -s <SE> option must be used to indicate which one of the replicas must be erased. The same is true if a GUID is specified, unless the --all-available option is used, in which case all replicas of the file will be deleted and unregistered (on a best-effort basis).

The following commands:

```
$ edg-rm --vo=dteam del guid:adb8e950-bf7e-11d7-a29c-fbbda1b7a6d1 -s wacdr002d.cern.ch
```

and

```
$ edg-rm --vo=dteam del guid:adb8e950-bf7e-11d7-a29c-fbbda1b7a6d1 --all-available
```

remove, from the file system and the catalog, one particular replica and all available replicas of the file, respectively.

6.2.2. Other Commands

Example 6.2.21 (Registering and unregistering Grid files)

Usually, new files are introduced in LCG-2 copying them from a non-grid resource using CopyAndRegisterFile; they are replicated to different SEs using replicateFile; and can be copied out of the Grid with copyFile. But it is also possible that a file is copied between SEs using copyFile (i.e., without registering) or by physically carrying a great amount of data in tapes, or it is possible that a new storage resource that already holds files is added to the Grid (becoming a SE). These files will be in a SE (they will have a valid SURL), but will not be registered in the LCG2 catalogs (i.e., they will not have an associated GUID).

For this situation, the registerFile (rf) and registerGUID (rg) commands may be useful. The registerFile command creates a new GUID for a given SURL, whereas registerGUID associates the replica identified by a SURL with an existent GUID (also specified as an argument). In the second case, it is assumed that there exist already some other replicas of the files that are being registered.

An example of the commands usage follows:

```
$ edg-rm --vo dteam rf sfn://tbed0101.cern.ch/flatfile/SE00/dteam/my_file1
> guid:c06a92ee-6911-11d8-a453-d9c1af867039

$ edg-rm --vo dteam rg sfn://wacdr002d.cern.ch/castor/cern.ch/grid/dteam/my_file3 \
guid:d3e9071e-687b-11d8-b3fa-8c0b6b5cbb30
> guid:d3e9071e-687b-11d8-b3fa-8c0b6b5cbb30
```

Likewise, instead of using the deleteFile, which both unregister and physically deletes a replica, a user can unregister a replica from the LRC catalogue, without actually deleting it (it can still be accessed on the SE with copyFile, for instance). This can be achieved with the unregisterFile (uf) command, specifying both the GUID and the SURL to be unregistered, as in:

```
$ edg-rm -i --vo=dteam unregisterFile guid:d3e9071e-687b-11d8-b3fa-8c0b6b5cbb30 \
sfn://wacdr002d.cern.ch/castor/cern.ch/grid/dteam/my_test3
```

If the last replica of a file is unregistered, then the GUI is also removed from the catalogue.

Example 6.2.22 (Managing aliases)

The addAlias (aa) command allows the user to add a new LFN to an existing GUID:

```
$ edg-rm --vo=dteam addAlias guid:c06a92ee-6911-11d8-a453-d9c1af867039 lfn:last_results
```

The removeAlias (ra) command allows the user to remove an LFN from an existing GUID:

```
$ edg-rm --vo=dteam ra guid:c06a92ee-6911-11d8-a453-d9c1af867039 lfn:last_results
```

In order to list the aliases of a file, the user has to use the edg-replica-metadata-catalog command, discussed later.

Example 6.2.23 (Listing an SE directory)

The list (ls) command can be used to list the contents of an SE directory (and, in the future, of an SRM directory):

```
$ edg-rm --vo dteam ls sfn://tbed0101.cern.ch/flatfile/SE00/dteam
> my_test1
> generated
> output.txt
> POOL-RM.txt
```

The argument of the command is a URI where the schema can be sfn, srm, or gsiftp.

6.2.3. Accessing a Grid File from a Job

As seen in Chapter 5, a job that is submitted to the Grid can access files stored in LCG-2. For that purpose, the JDL file of the job must include the name (GUID or LFN) of the files to be accessed, in the InputData attribute; and the protocol that will be used to access them. Currently, the only two supported protocols to access grid files are: GridFTP (gsiftp) and rfio (rfio).

The following examples show access of the files from a perl script. It could be done also from a C++ or Java program, using the respective edg-rm and rfio APIs. For information on that, please refer to [R25] and [R26].

Example 6.2.31 (Accessing a file using the GridFTP protocol)

We assume that a user has registered a data file (called values) within LCG-2, using lfn:example_values as its LFN. The contents of the file are the following:

```
 The contents of these lines,
which are not really important,
will be shown in the std.out file.
```

The JDL file of the job (example.jdl) includes the LFN of the file, and the protocol (gsiftp) to be used when accessing it. The contains of the JDL file follows:

```
 Executable="example.pl";
StdOutput="std.out";
StdError="std.err";
InputSandbox="example.pl";
OutputSandbox="std.out","std.err";
InputData="lfn:example_values";
DataAccessProtocol="gsiftp";
```

The executable (example.pl) is a perl program, that calls the edg-rm copyFile command (already explained) to copy the grid file to the local filesystem of the Worker Node where the job is running. The rest of the script is simple perl code to show the data retrieved:

```
#!/usr/bin/perl
        # Copy the input data file to the WN local filesystem
system "edg-rm --vo=dteam copyFile lfn:example_values file: `pwd `/values";
        # Open it
open(file,'values');
        # Read all the lines
@lines=<file>;
        #Show the info
print "The values stored in the input data file are:\n";
print " @lines";
```

The job is submitted as usual:

```
 $ edg-job-submit -o jobid example.jdl
```

And the results retrieved with:

```
 $ edg-job-get-output -i jobid
```

The std.out file obtained is this:

```
The values stored in the input data file are:
  The contents of these lines,
  which are not really important,
  will be shown in the std.out file.
```

Example 6.2.32 (Accessing a file using the rfio protocol)

This example is very similar to the previous one, but here the rfio protocol is used. The same data file values is used, and the only changes in the new example2.jdl file are the executable file, and the access protocol:

```
 Executable="example2.pl";
StdOutput="std.out";
StdError="std.err";
InputSandbox="example2.pl";
OutputSandbox="std.out","std.err";
InputData="lfn:example_values";
DataAccessProtocol="rfio";
```

The example2.pl file is a bit more complicated this time, because the rfio protocol cannot handle LFNs, and needs the complete path to the file instead. For this reason, the TURL of the file is obtained first and then it is adapted to rfio needs. The commands to get the TURL from a known LFN have been already seen and could be also performed manually instead of inserting them in the perl script, but are included here for completeness. The form of the TURL will be: rfio://<hostname>/<path>, while the rfio command expects a <hostname>:<path> string, and therefore the perl code has to do a little extra work to adapt the string before invoking the rfcp command, which copies the file to the WN local filesystem.

```
#!/usr/bin/perl
        # Obtain the SURL of the file whose LFN we know
$surl= `edg-rm --vo dteam lr lfn:example_values `;
chop($surl);
        # Now obtain the TURL for the rfio protocol
$turl= `edg-rm --vo dteam getTurl $surl rfio `;
        # Adapt the "rfio://hostname/path" form returned to the "hostname:path" form that rfio uses
$turl= s/rfio:\/\///;                   # delete the extra "//"
$turl= s/\//:\//;                       # add the ":"
chop($turl);
        # Copy the input data file to the WN local filesystem
system "rfcp $turl    `pwd `/values";
        # Open it
open(file,'values');
        # Read all the lines
@lines=<file>;
        #Show the info
print "The values stored in the input data file are:\n";
print " @lines";
```

The job is submitted and the output retrieved like in the previous example, and the retrieved std.out file is:

```
95 bytes in 0 seconds through eth0 (in) and local (out)
The values stored in the input data file are:
  The contents of these lines,
  which are not really important,
  will be shown in the std.out file.
```

where the first line is produced by the rfcp command.

6.3. EDG-LRC AND EDG-RMC CLIENT TOOLS

The edg-local-replica-catalog and edg-replica-metadata-catalog client tools are low level tools that allow users to browse and directly manipulating the LRC and the RMC catalogues.

Attention! *With these tools, a user can change the content of the catalogues making them inconsistent. For instance, a GUID can be removed from the RMC but not from the LRC making a file not addressable by its alias. In normal operation, a user should preferably use the edg-replica-manager client tools, and only use these ones with extreme care.*

The edg-lrc and edg-rmc commads follow the same syntax as those of edg-rm. First some general options can be specified, they are followed by a particular command name with its arguments, and finally the specific command options are given.

NOTE: *If the above described order is not followed, the general and particular options may be mixed, resulting in a fail of the command.*

Important: *For the reasons explained in Section 6.5, the LRC and RMC catalogs do not store the guid: and lfn: prefixes for GUIDs and LFNs*[5]*. They only store the string following those prefixes. Currently, when querying for a GUID or a LFN using edg-lrc or edg-rmc, the prefixes must not be used (opposite to what happens with edg-rm). However, the prefixes have to be used when manually adding new catalogs mappings, as the syntax checking enforces it, and these prefixes are also stored in the catalog. Unfortunately, this leads to catalog entries* ***without*** *prefixes when the registration is automatic (i.e., when using edg-rm) and* ***with*** *them, when it is manual (with edg-lrc or edg-rmc); as will be later shown in the examples. This behaviour may be changed in future releases, but it reinforces the idea that these tools should only be used by system administrators and not by final users.*

Only some usage examples of the most important commands will be given here. For detailed information please refer to [R20] and [R21].

6.3.1. Local Replica Catalog Commands

The edg-lrc commands operate with GUID-SURLs mappings. ***Note:*** *In the commands name and in the manpages, the SURL is often called PFN (for Physical File Name). The -i option is used to connect to the LRC using http instead of https (sometimes it may be the only available way to connect to the server).*

All the commands require the LRC ***endpoint****, which can be obtained using the edg-rm printInfo command. This usually takes the form:*

http(s)://<hostname>:<port>/<VO>/edg-local-replica-catalog/services/edg-local-replica-catalog

*It can be specified either using the --endpoint option followed by the full endpoint, or setting the values for the hostname, the port and the VO to be used, with the -h, -p and --vo options respectively (****Note:*** *It is safer to use the --endpoint option, since it does not make any assumption regarding the path).*

The following tables summarize the most useful commands;

[5] They do store the prefix for SURLs (*sfn:*)

Mapping management commands:

addMapping guid pfn	*Add the given mapping to the catalog.*
pfnExists pfn	*Does the PFN exist in this catalog?*
guidExists guid	*Does the GUID exist in this catalog?*
guidForPfn pfn	*Return the GUID for a given PFN.*
pfnsForGuid guid	*Return the PFNs for a given GUID.*
removePfn guid pfn	*Remove a PFN from a given GUID.*

Wildcard query commands (to retrieve GUIDs or SURLs that match a pattern):

mappingsByPfn pfnPattern	*Get a set of mappings by a wildcard search on PFN name.*
mappingsByGuid guidPattern	*Get a set of mappings by a wildcard search on guid.*
getResultLength	*Return the result length by default (i.e. how many mappings will be returned when using mappingsByGuid or mappingsByPfn).*
setResultLength length	*Set the result length by default (i.e. how many mappings will be returned when using mappingsByGuid or mappingsByPfn).*

There are also some other commands to set/get different attributes of the GUID-PFN mappings, or to retrieve mappings whose attribute satisfy certain conditions. For details refer to [R20].

Examples.

For clarity reasons, environmental variables are used in the following examples, instead of long file names. Thus, it will be assumed that a file is registered in the Grid with its GUID, SURL and LFN assigned to:

```
$ setenv GUID c06a92ee-6911-11d8-a453-d9c1af867039
$ setenv SURL sfn://tbed0101.cern.ch/flatfile/SE00/dteam/my_test1
$ setenv ALIAS lasts_results
```

In addition, some false values (not assigned to any real file) are defined:

```
$ setenv GUID2 c06a92ee-6911-11d8-a453-000000000000
$ setenv SURL2 sfn://tbed0101.cern.ch/flatfile/SE00/dteam/my_fake
$ setenv ALIAS2 fake_alias
```

Example 6.3.11 (Checking existence of SURLs and GUIDs)

Confirming that $SURL and $GUID exist, but $SURL2 does not:

```
$ edg-lrc pfnExists $SURL --endpoint \
http://rlscert01.cern.ch:7777/dteam/v2.2/edg-local-replica-catalog/services/edg-local-replica-catalog
> Pfn exists : 'sfn://tbed0101.cern.ch/flatfile/SE00/dteam/my_test1'

$ edg-lrc guidExists $GUID --endpoint \
http://rlscert01.cern.ch:7777/dteam/v2.2/edg-local-replica-catalog/services/edg-local-replica-catalog
> GUID exists : 'c06a92ee-6911-11d8-a453-d9c1af867039'
```

*$ edg-lrc pfnExists $SURL2 - -endpoint *
http://rlscert01.cern.ch:7777/dteam/v2.2/edg-local-replica-catalog/services/edg-local-replica-catalog
> Pfn does not exist : 'sfn://tbed0101.cern.ch/flatfile/SE00/dteam/my_fake'

Example 6.3.12 (Retrieving SURLs and GUIDs)

Retrieving the GUID for a SURL.

*$ edg-lrc guidForPfn $SURL - -endpoint *
http://rlscert01.cern.ch:7777/dteam/v2.2/edg-local-replica-catalog/services /edg-local-replica-catalog
> c06a92ee-6911-11d8-a453-d9c1af867039

Retrieving the SURLs for a GUID (if it exists):

*$ edg-lrc pfnsForGuid $GUID - -endpoint *
http://rlscert01.cern.ch:7777/dteam/v2.2/edg-local-replica-catalog /services/edg-local-replica-catalog
> sfn://tbed0101.cern.ch/flatfile/SE00/dteam/my_test1

*$ edg-lrc pfnsForGuid $GUID2 - -endpoint *
http://rlscert01.cern.ch:7777/dteam/v2.2/edg-local-replica-catalog/services /edg-local-replica-catalog
> No such guid : 'c06a92ee-6911-11d8-a453-00000000000'

Example 6.3.13 (Retrieving with wildcards)

Retrieving GUIDs for a SURL pattern:

*$ edg-lrc mappingsByPfn '*my_test*' - -endpoint *
http://rlscert01.cern.ch:7777/dteam/v2.2/edg-local-replica-catalog/services /edg-local-replica-catalog
> d3e9071e-687b-11d8-b3fa-8c0b6b5cbb30, sfn://wacdr002d.cern.ch/castor/cern.ch/grid/dteam/my_test3
> c06a92ee-6911-11d8-a453-d9c1af867039, sfn://tbed0101.cern.ch/flatfile/SE00/dteam/my_test1

Retrieving SURLs for a GUID pattern:

*$ edg-lrc mappingsByGuid '*b3fa*' - -endpoint *
http://rlscert01.cern.ch:7777/dteam/v2.2/edg-local-replica-catalog/services /edg-local-replica-catalog
> guid:0abdd087-5a43-11d8-b57f-a48b3faf9ccd, sfn://lxshare0291.cern.ch/flatfiles/LCG-CERT-SE03
/dteam/generated/ 2004/02/08/file05e657d6-5a43-11d8-b57f-a48b3faf9ccd
> guid:0abdd087-5a43-11d8-b57f-a48b3faf9ccd, sfn://lxshare0236.cern.ch/flatfiles/LCG-CERT-SE01/
dteam/generated/2004/02/08/file1010ba9e-5a43-11d8-9971-fa6a704d33db
> d3e9071e-687b-11d8-b3fa-8c0b6b5cbb30, sfn://wacdr002d.cern.ch/castor/cern.ch/grid/dteam/my_test3
. . .

Example 6.3.14 (Adding a mapping)

Adding a mapping with a false SURL (note that when inserting a mapping the guid: prefix must be used):

```
$ edg-lrc addMapping guid:$GUID $SURL2 - -endpoint \
http://rlscert01.cern.ch:7777/dteam/v2.2/edg-local-replica-catalog/services > /edg-local-replica-catalog

$ edg-lrc pfnExists $SURL2 - -endpoint \
http://rlscert01.cern.ch:7777/dteam/v2.2/edg-local-replica-catalog/serv ices/edg-local-replica-catalog
> Pfn exists : 'sfn://tbed0101.cern.ch/flatfile/SE00/dteam/my_fake
```

Example 6.3.15 (Removing a mapping)

Removing the previously added SURL:

```
$ edg-lrc removePfn guid:$GUID $SURL2 - -endpoint \
http://rlscert01.cern.ch:7777/dteam/v2.2/edg-local-replica-catalog/services/edg-local-replica-catalog
```

6.3.2. Replica Metadata Catalog Commands

The edg-rmc commands operate with GUID-LFNs mappings. The -i option is used in the same way as with edg-lrc, and so are the options used to specify the endpoint for the RMC server (which, again, can be obtained with the edg-rm printInfo command).

The following tables summarize the most useful commands;

Mapping management commands:

addAlias guid alias	*Add a new alias to the catalog.*
aliasExists alias	*Does the alias exist in this catalog?*
guidExists guid	*Does the GUID exist in this catalog?*
guidForAlias alias	*Return the GUID for a given alias.*
aliasesForGuid guid	*Return the aliases for a given GUID.*
removeAlias guid alias	*Remove an alias from a given GUID.*

Wildcard query commands (to retrieve GUIDs or SURLs that match a pattern):

mappingsByAlias aliasPattern	*Get a set of mappings by a wildcard search on alias name.*
mappingsByGuid guidPattern	*Get a set of mappings by a wildcard search on guid.*
getResultLength	*Return the result length by default (i.e. how many mappings will be returned when using mappingsByGuid or mappingsByAlias).*
setResultLength length	*Set the result length by default (i.e. how many mappings will be returned when using mappingsByGuid or mappingsByAlias).*

As in the case of edg-lrc, there are some other commands that set/get attributes for the GUIDs or the aliases, and some that retrieve mappings whose attribute satisfy certain conditions. For details refer to [R21].

Examples.

The same environmental variables of the previous section are used in the following examples.

Example 6.3.21 (Checking the existence of GUIDs and LFNs)

Confirming that $ALIAS exists but $ALIAS2 does not.

$ edg-rmc aliasExists $ALIAS - -endpoint http://rlscert01.cern.ch:7777/dteam/v2.2/edg-replica-metadata-catalog/
services/edg-replica-metadata-catalog
> Alias exists : 'last_results'

$ edg-rmc guidForAlias $ALIAS2 - -endpoint http://rlscert01.cern.ch:7777/dteam/v2.2/edg-replica-metadata-catalog/services/edg-replica-metadata-catalog
> No such alias : 'lfn:fake_alias'

The same for $GUID and $GUID2.

$ edg-rmc guidExists $GUID - -endpoint http://rlscert01.cern.ch:7777/dteam/v2.2/edg-replica-metadata-catalog/services/edg-replica-metadata-catalog
> GUID exists : 'c06a92ee-6911-11d8-a453-d9c1af867039'

$ edg-rmc guidExists $GUID2 - -endpoint http://rlscert01.cern.ch:7777/dteam/v2.2/edg-replica-metadata-catalog/services/edg-replica-metadata-catalog
> GUID does not exist : 'c06a92ee-6911-11d8-a453-00000000000'

Example 6.3.22 (Retrieving LFNs and GUIDs)

Retrieving the GUID for a known alias.

$ edg-rmc guidForAlias $ALIAS - -endpoint http://rlscert01.cern.ch:7777/dteam/v2.2/edg-replica-metadata-catalog/services/edg-replica-metadata-catalog
> c06a92ee-6911-11d8-a453-d9c1af867039

Retrieving the existent aliases for a GUID.

$ edg-rmc aliasesForGuid $GUID - -endpoint http://rlscert01.cern.ch:7777/dteam/v2.2/edg-replica-metadata-catalog/services/edg-replica-metadata-catalog
> last_results

Example 6.3.23 (Adding new LFNs)

In order to add a new alias, the guid: and lfn: prefixes must be used. Consider the following example, where only the last command is accepted:

$ edg-rmc addAlias $GUID lfn:new_results - -endpoint http://rlscert01.cern.ch:7777/dteam/v2.2/edg-replica-metadata-catalog/services/edg-replica-metadata-catalog
> Error: addAlias: Invalid file type for URI : 'c06a92ee-6911-11d8-a453-d9c1af867039', reason : Scheme is not 'guid'

$ edg-rmc addAlias guid:$GUID new_results - -endpoint http://rlscert01.cern.ch:7777/dteam/v2.2/edg-replica-metadata-catalog/services/edg-replica-metadata-catalog
> Error: addAlias: Invalid file type for URI : 'new_results', reason : Scheme is not 'lfn'

$ edg-rmc addAlias guid:$GUID lfn:new_results - -endpoint http://rlscert01.cern.ch:7777/dteam/v2.2/edg-replica-metadata-catalog/services/edg-replica-metadata-catalog

Example 6.3.24 (Retrieving with wildcards)

Using an alias pattern, two mappings are returned. One corresponds to the real stored file, and the other one (with prefixes) was created in the previous example. Note that the new alias cannot be added to the existent (without prefix) GUID. For that purpose the edg-rm addAlias command can be used.

*$ edg-rmc mappingsByAlias '*result*' - -endpoint http://rlscert01.cern.ch:7777/dteam/v2.2/edg-replica-metadata-catalog/services/edg-replica-metadata-catalog*
> c06a92ee-6911-11d8-a453-d9c1af867039, last_results
> guid:c06a92ee-6911-11d8-a453-d9c1af867039, lfn:new_results

A GUID pattern can also be used:

$ edg-rmc mappingsByGuid $GUID - -endpoint http://rlscert01.cern.ch:7777/dteam/v2.2/edg-replica-metadata-catalog/services/edg-replica-metadata-catalog
> c06a92ee-6911-11d8-a453-d9c1af867039, last_results

Example 6.3.25 (Deleting an LFN)

The previously added mapping is removed:

$ edg-rmc removeAlias guid:$GUID lfn:new_results - -endpoint http://rlscert01.cern.ch:7777/dteam/v2.2/edg-replica-metadata-catalog/services > /edg-replica-metadata-catalog

6.4. LOW LEVEL DATA MANAGEMENT TOOLS

The low level tools allow users to perform some actions on the GridFTP server of a SE. A brief summary of their functions follow:

edg-gridftp-exists URL	*Check the existence of a file or directory on a SE.*
edg-gridftp-ls URL	*List a directory on a SE.*
edg-gridftp-mkdir URL	*Create a directory on a SE.*
edg-gridftp-rename sourceURL destURL	*Rename a file on a SE.*
edg-gridftp-rmdir URL	*Remove a directory on a SE.*
globus-url-copy sourceURL destURL	*Copy files between SEs.*

The commands edg-gridftp-rename, edg-gridftp-rm, and edg-gridftp-rmdir should be used with extreme care and only in case of serious problems. In fact these commands do not interact with any of the catalogues and therefore they can compromise the consistency/coherence of the information contained.

To obtain help on these commands use the option --usage or --help. General information on GridFTP is available in [R7].

6.5. POOL AND LCG-2

*The **POOL** tool (**P**OOL **O**f persistent **O**bjects for **L**HC) is used by most of the LHC experiments as a common persistency framework for the LCG application area. Objects created by users using POOL are stored into its own File Catalog (XML Catalog). This File Catalog keeps track of all POOL databases and resolves file references into PFN which are then used by lower level components like the storage service to access file contents.*

Until now, the POOL catalog (XML) and the EDG Replica location Service (RLS) were working in parallel. This could cause therefore a lack of communication between files created and registered into the XML catalog with those files registered into the RLS. The new LCG-2 release has observed such problem and has update its software to make entries in XML and RLS compatible.

6.5.1. LCG Catalog (RLS) vs POOL Catalog (XML)

One problem appears to make the RLS and the XML catalogs compatible:

- *The EDG Replica Manager writes entries in the RLS for LFNs, GUIDs and SURLs giving explicitly the prefix of the entry. For example, these are typical RLS entries:*

  LFN - lfn:this-is-my-logical-file-name
  GUID - guid:73e16e74-26b0-11d7-b1e0-c5c68d88236a
  SURL - sfn://lxshare0384.cern.ch/flatfiles/cms/data/05/x.dat

- *However, POOL stores the same entries as follows:*

  LFN - this-is-my-logical-file-name
  GUID - 73e16e74-26b0-11d7-b1e0-c5c68d88236a
  SURL - sfn://lxshare0384.cern.ch/flatfiles/cms/data/05/x.dat
  - any format accepted by ROOT
  ex.: /home/user/mydata/file.dat
  rfio:/afs/cern.ch/project/cms/data/file.dat
  d-cache:/pool/disks/cms/file.dat

The problem was therefore that an entry inserted by POOL in the RLS cannot be processed by the EDG RM and viceversa. LCG-2 solved this problem changing the EDG Replica Manager to store LFNs and GUIDs as POOL does (i.e., without the guid: and lfn: prefixes).

Example 6.5.11 (Migration from POOL(XML) to LCG(RLS))

We assume that the user has used POOL and as result has created a file which has been registered into the XML catalog of POOL. Now the point is how to register this file into the LCG catalog, the RLS.

- *Basically it is just necessary to obtain the connection to the RLS catalog. A contact string has to be specified through the environment variable POOL_CATALOG as follows:*

  $ export POOL_CATALOG=edgcatalog_http://<host>:<port>/<path> (in the case of sh shell)

  $ setenv POOL_CATALOG edgcatalog_http://<host>:<port>/<path> (in the case of csh shell)

  For example into LCG-2 this environment variable should be set to:

  $ export POOL_CATALOG=edgcatalog_http://rlscert01.cern.ch:7777/$VO/v2.2/edg-local-replica-catalog/services/edg-local-replica-catalog

- *In the case that the user has specified the file as a SURL into POOL, he can assign it a LFN with POOL as follows:*

  $ FCregisterLFN -p <SURL> -l <LFN>

- *Now the user can make some test to probe whether the file is into the LRC with the RLS client:*

  $ edg-lrc mappingsByPfn <SURL> --endpoint <LRC>

- *Or into the RMC:*

  $ edg-rmc mappingsByAlias <LFN> --endpoint <RMC>

- *Finally he can check if the RM is able to find the file:*

  $ edg-rm --vo=<VO> lr lfn:<LFN>

- *note that in case the POOL user has defined the SURL entry following a ROOT format, he must use the command FCrenamePFN to create a SURL entry compatible with the RLS catalog.*

A complete list of POOL commands can be found into [R27]. The user can see them just by typing FC<tab>.

7. INFORMATION SYSTEM

*In the following sections examples are given on how to interrogate the Information System in LCG-2 Grid. In particular, the different servers from which the information can be obtained are discussed. These are the local GRISes, the site GIISes and the global BDIIs. As explained earlier, the data in the IS of LCG-2 conforms to the GLUE Schema. For a list of GLUE Schema elements (**objectclasses**) and their attributes, check Appendix A.*

7.1. THE LOCAL GRIS

The local GRISes running on Computing Elements and Storage Elements at the different sites report information on the characteristics and status of the services. They give both static and dynamic information.

In order to interrogate the GRIS on a specific Grid Element, the hostname of the Grid Element and the TCP port where the GRIS run must be specified. Such port is always 2135. The following command can be used:

```
$ ldapsearch -x -h <hostname> -p 2135 -b "mds-vo-name=local, o=grid"
```

where the -x option indicates that simple authentication (instead of LDAP's SASL) should be used; the -h and -p options precede the hostname and port respectively; and the -b option is used to specify the initial search node in the LDAP tree.

The same effect can be obtained with:

```
$ ldapsearch -x -H <LDAP_URI> -b "mds-vo-name=local, o=grid"
```

where the hostname and port are included in the -H <LDAP_URI> option, avoiding the use of -h and -p.

Example 7.11 (Interrogating the GRIS on a Computing Element)

The command used to interrogate the GRIS located on host lxn1181 is:

```
$ ldapsearch -x -h lxn1181.cern.ch -p 2135 -b "mds-vo-name=local, o=grid"
```

or:

```
$ ldapsearch -x -H ldap://lxn1181.cern.ch:2135 -b "mds-vo-name=local, o=grid"
```

And the obtained reply will be:

```
version: 2

#
# filter: (objectclass=*)
# requesting: ALL
#
```

 # lxn1181.cern.ch/siteinfo, local, grid
dn: in=lxn1181.cern.ch/siteinfo,Mds-Vo-name=local,o=grid
objectClass: SiteInfo
objectClass: DataGridTop
objectClass: DynamicObject
siteName: CERN-LCG2
sysAdminContact: hep-project-grid-cern-testbed-managers@cern.ch
userSupportContact: hep-project-grid-cern-testbed-managers@cern.ch
siteSecurityContact: hep-project-grid-cern-testbed-managers@cern.ch
dataGridVersion: LCG-2_0_0beta
installationDate: 20040106120000Z

 # lxn1181.cern.ch:2119/jobmanager-lcgpbs-infinite, local, grid
dn: GlueCEUniqueID=lxn1181.cern.ch:2119/jobmanager-lcgpbs-infinite, mds-vo-name=local, o=grid
objectClass: GlueCETop
objectClass: GlueCE
objectClass: GlueSchemaVersion
objectClass: GlueCEAccessControlBase
objectClass: GlueCEInfo
objectClass: GlueCEPolicy
objectClass: GlueCEState
objectClass: GlueInformationService
objectClass: GlueKey
GlueSchemaVersionMajor: 1
GlueSchemaVersionMinor: 1
GlueCEName: infinite
GlueCEUniqueID: lxn1181.cern.ch:2119/jobmanager-lcgpbs-infinite
GlueCEInfoGatekeeperPort: 2119
GlueCEInfoHostName: lxn1181.cern.ch
GlueCEInfoLRMSType: pbs
GlueCEInfoLRMSVersion: OpenPBS_2.4
GlueCEInfoTotalCPUs: 16
GlueCEStateEstimatedResponseTime: 0
GlueCEStateFreeCPUs: 16
GlueCEStateRunningJobs: 0
GlueCEStateStatus: Production
GlueCEStateTotalJobs: 0
GlueCEStateWaitingJobs: 0
GlueCEStateWorstResponseTime: 0
GlueCEPolicyMaxCPUTime: 172800
GlueCEPolicyMaxRunningJobs: 99999
GlueCEPolicyMaxTotalJobs: 999999
GlueCEPolicyMaxWallClockTime: 259200
GlueCEPolicyPriority: 1
GlueCEAccessControlBaseRule: VO:alice
GlueCEAccessControlBaseRule: VO:atlas
GlueCEAccessControlBaseRule: VO:cms
GlueCEAccessControlBaseRule: VO:lhcb
GlueCEAccessControlBaseRule: VO:dteam

GlueForeignKey: GlueClusterUniqueID=lxn1181.cern.ch
GlueInformationServiceURL: ldap://lxn1181.cern.ch:2135/mds-vo-name=local,o=grid
. . .

In order to restrict the search to a specific objectclass, a filter of the form 'objectclass=<name>' can be used. By specifying a list of attribute names, the reply is limited to the value of those attributes for the corresponding objectclass, as is shown in the next example. A description of all objectclasses and their attributes to optimize the LDAP search command can be found in Appendix A.

Example 7.12 (Getting information about the site name from the GRIS on a Computing Element)

```
$ ldapsearch -x -h lxn1181.cern.ch -p 2135 -b "mds-vo-name=local, o=grid" 'objectclass=SiteInfo'
siteName
version: 2

#
# filter: objectclass=SiteInfo
# requesting: siteName
#

# lxn1181.cern.ch/siteinfo, local, grid
dn: in=lxn1181.cern.ch/siteinfo,Mds-Vo-name=local,o=grid
siteName: CERN-LCG2

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

By adding the -LLL option we can avoid the comments and the version information in the reply.

```
$ ldapsearch -LLL -x -h lxn1181.cern.ch -p 2135 -b "mds-vo-name=local, o=grid" 'objectclass=SiteInfo'
siteName

dn: in=lxn1181.cern.ch/siteinfo,Mds-Vo-name=local,o=grid
siteName: CERN-LCG2
```

7.2. THE SITE GIIS

At each site, a site GIIS collects information about all resources present at a site (i.e. data from all GRISes of the site).

For a list of all sites and all resources present, please refer to:

http://grid-deployment.web.cern.ch/grid-deployment/cgi-bin/index.cgi?var=gis/lcg2Status

Usually a site GIIS runs on a Computing Element. In order to interrogate the site GIIS for FZK, one needs to find out the name of that CE. This can be found in the web page reporting the site status: http://lcg.fzk.de/lcg/lcg2.html

The port used to interrogate a site GIIS is usually the same as that of GRISes: 2135. In order to interrogate the GIIS (and not the local GRIS) a different base name must be used (instead of mds-vo-name=local, o=grid). This base name is just the site name, which is published by all sites, where all "-" characters have been removed. So, for instance, for FZK, the site name is FZK-LCG2 and the mds base name is mds-vo-name=fzklcg2, o=grid.

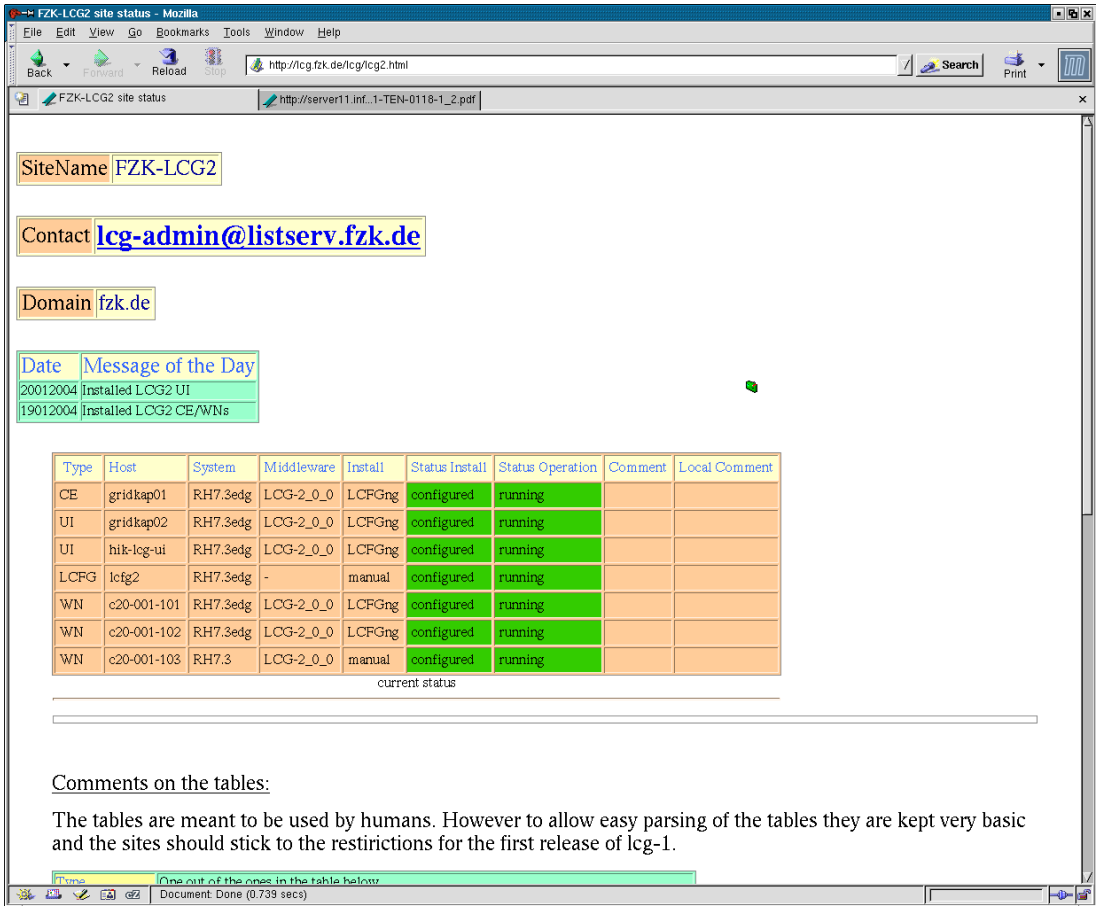

FZK-LCG2 site status - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://lcg.fzk.de/lcg/lcg2.html
Search Print
FZK-LCG2 site status
http://server11.inf...1-TEN-0118-1_2.pdf
SiteName FZK-LCG2
Contact lcg-admin@listserv.fzk.de
Domain fzk.de
Date Message of the Day
20012004 Installed LCG2 UI
19012004 Installed LCG2 CE/WNs
Type Host System Middleware Install Status Install Status Operation Comment Local Comment
CE gridkap01 RH7.3edg LCG-2_0_0 LCFGng configured running
UI gridkap02 RH7.3edg LCG-2_0_0 LCFGng configured running
UI hik-lcg-ui RH7.3edg LCG-2_0_0 LCFGng configured running
LCFG lcfg2 RH7.3edg - manual configured running
WN c20-001-101 RH7.3edg LCG-2_0_0 LCFGng configured running
WN c20-001-102 RH7.3edg LCG-2_0_0 LCFGng configured running
WN c20-001-103 RH7.3 LCG-2_0_0 manual configured running
current status
Comments on the tables:
The tables are meant to be used by humans. However to allow easy parsing of the tables they are kept very basic and the sites should stick to the restirictions for the first release of lcg-1.
Document: Done (0.739 secs)

Figure 8: The status page of the FZK site

As we can see in Figure 8, the CE name is gridkap01.fzk.de.

So, in order to interrogate the site GIIS, we can use the command shown in the following example:

Example 7.21 (Interrogating the site GIIS)

$ldapsearch -x -H ldap://gridkap01.fzk.de:2135 -b "mds-vo-name=fzklcg2,o=grid" version: 2

#
 # filter: (objectclass=)*
 # requesting: ALL
#

 # gridkap01.fzk.de/siteinfo, fzklcg2, grid
dn: in=gridkap01.fzk.de/siteinfo,Mds-Vo-name=fzklcg2,o=grid
objectClass: SiteInfo
objectClass: DataGridTop
objectClass: DynamicObject
siteName: FZK-LCG2
sysAdminContact: lcg-admin@listserv.fzk.de
userSupportContact: lcg-admin@listserv.fzk.de
siteSecurityContact: lcg-admin@listserv.fzk.de
dataGridVersion: LCG-2_0_0beta
installationDate: 20040119103800Z

 # gridkap01.fzk.de:2119/jobmanager-lcgpbs-long, fzklcg2, grid
dn: GlueCEUniqueID=gridkap01.fzk.de:2119/jobmanager-lcgpbs-long, Mds-Vo-name=fzklcg2,o=grid
objectClass: GlueCETop
objectClass: GlueCE
objectClass: GlueSchemaVersion
objectClass: GlueCEAccessControlBase
objectClass: GlueCEInfo
objectClass: GlueCEPolicy
objectClass: GlueCEState
objectClass: GlueInformationService
objectClass: GlueKey
GlueSchemaVersionMajor: 1
GlueSchemaVersionMinor: 1
GlueCEName: long
GlueCEUniqueID: gridkap01.fzk.de:2119/jobmanager-lcgpbs-long
GlueCEInfoGatekeeperPort: 2119
GlueCEInfoHostName: gridkap01.fzk.de
GlueCEInfoLRMSType: pbs
. . .
 # gridkap01.fzk.de:2119/jobmanager-lcgpbs-infinite, fzklcg2, grid
dn: GlueCEUniqueID=gridkap01.fzk.de:2119/jobmanager-lcgpbs-infinite, Mds-Vo-name=fzklcg2,o=grid
objectClass: GlueCETop
objectClass: GlueCE
objectClass: GlueSchemaVersion
objectClass: GlueCEAccessControlBase
objectClass: GlueCEInfo
objectClass: GlueCEPolicy
objectClass: GlueCEState
objectClass: GlueInformationService
objectClass: GlueKey

GlueSchemaVersionMajor: 1
GlueSchemaVersionMinor: 1
GlueCEName: infinite
GlueCEUniqueID: gridkap01.fzk.de:2119/jobmanager-lcgpbs-infinite
. . .
 # rfio, gridkap02.fzk.de, fzklcg2, grid
dn: GlueSEAccessProtocolType=rfio,GlueSEUniqueID=gridkap02.fzk.de,Mds-Vo-name=fzklcg2,o=grid
objectClass: GlueSETop
objectClass: GlueSEAccessProtocol
objectClass: Gluekey
objectClass: GlueSchemaVersion
GlueSEAccessProtocolType: rfio
GlueSEAccessProtocolPort: 5001
GlueChunkKey: GlueSEUniqueID=gridkap02.fzk.de
GlueSchemaVersionMajor: 1
GlueSchemaVersionMinor: 1

 # gridkap02.fzk.de, fzklcg2, grid
dn: GlueSLUniqueID=gridkap02.fzk.de,Mds-Vo-name=fzklcg2,o=grid
objectClass: GlueSLTop
objectClass: GlueSL
objectClass: GlueSLArchitecture
objectClass: Gluekey
objectClass: GlueSchemaVersion
GlueSLUniqueID: gridkap02.fzk.de
GlueSLName: FZK-LCG2
GlueSLArchitectureType: disk
GlueForeignKey: GlueSEUniqueID=gridkap02.fzk.de
GlueSchemaVersionMajor: 1
GlueSchemaVersionMinor: 1
. . .

7.3. THE BDII

Each site running a Resource Broker runs as well a BDII that collects all information coming from the Regional GIISes and stores them in a permanent database. In order to find out the location of the BDII you can consult the web page of the LCG-2 site status as done for the site GIISes.

The BDII can be interrogated using the standard mds base: mds-vo-name=local, o=grid, and the BDII port: 2170.

Example 7.31 (Interrogating a BDII)

In this example, two attributes from the GlueCESEBind object class are retrieved for all sites.

*$ ldapsearch -x -LLL -H ldap://lxshare0222.cern.ch:2170 -b "mds-vo-name=local,o=grid" *

'objectclass=GlueCESEBind' GlueCESEBindCEUniqueID GlueCESEBindSEUniqueID

dn: GlueCESEBindSEUniqueID=grid100.kfki.hu, GlueCESEBindGroupCEUniqueID=grid109.kfki.hu:2119
/jobmanager-pbs-infinite, Mds-Vo-name=budapestlcg1, Mds-Vo-name=lcgeast,Mds-Vo-name=local,o=grid
GlueCESEBindCEUniqueID: grid109.kfki.hu:2119/jobmanager-pbs-infinite
GlueCESEBindSEUniqueID: grid100.kfki.hu

dn: GlueCESEBindSEUniqueID=grid100.kfki.hu, GlueCESEBindGroupCEUniqueID=grid109.kfki.hu:2119
/jobmanager-pbs-long, Mds-Vo-name=budapestlcg1,Mds-Vo-name=lcgeast,Mds-Vo-name=local,o=grid
GlueCESEBindCEUniqueID: grid109.kfki.hu:2119/jobmanager-pbs-long
GlueCESEBindSEUniqueID: grid100.kfki.hu

dn: GlueCESEBindSEUniqueID=grid100.kfki.hu, GlueCESEBindGroupCEUniqueID=grid109.kfki.hu:2119
/jobmanager-pbs-short, Mds-Vo-name=budapestlcg1,Mds-Vo-name=lcgeast,Mds-Vo-name=local,o=grid
GlueCESEBindCEUniqueID: grid109.kfki.hu:2119/jobmanager-pbs-short
GlueCESEBindSEUniqueID: grid100.kfki.hu

dn: GlueCESEBindSEUniqueID=adc0021.cern.ch, GlueCESEBindGroupCEUniqueID=adc0015.cern.ch:
2119/jobmanager-lcgpbs-infinite, Mds-Vo-name=cernlcg1,Mds-Vo-name=lcgeast,Mds-Vo-name=local,o=grid
GlueCESEBindCEUniqueID: adc0015.cern.ch:2119/jobmanager-lcgpbs-infinite
. . .

Example 7.32 (Listing all the CEs which publish a given tag querying the BDII)

*The attribute GlueHostApplicationSoftwareRunTimeEnvironment can be used to publish experiment-specific information (**tag**) on a CE, for example that a given experiment software is installed. To list all the CEs which publish a given tag, a query to the BDII can be performed. In this example, that information is retrieved for all the subclusters:*

```
$ ldapsearch -h lxshare0222.cern.ch -p 2170 \
-b "mds-vo-name=local,o=grid" -x 'objectclass=GlueSubCluster' \
GlueChunkKey GlueHostApplicationSoftwareRunTimeEnvironment
```

Example 7.33 (Listing all the SEs which support a given VO)

A storage element supports a VO if users of that VO are allowed to store files on that SE. It is possible to find out which SEs support a VO with a query to the BDII. For example, to have the list af all SEs supporting ATLAS, the GlueSAAccessControlBaseRule, which specifies a supported VO, is used:

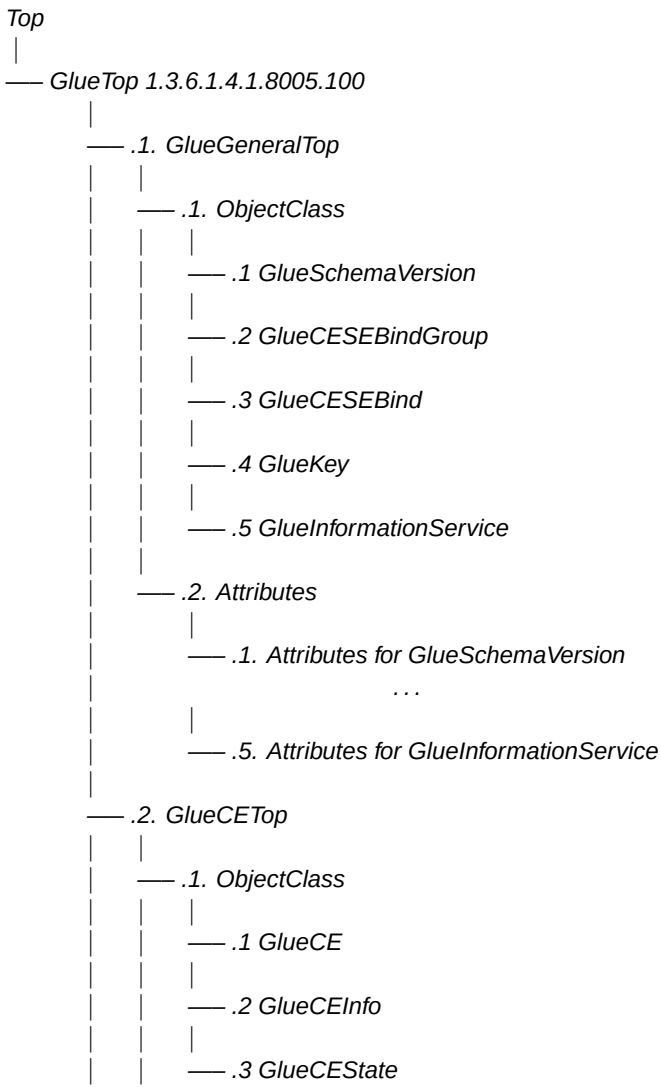
```
$ ldapsearch -h lxshare0222.cern.ch -p 2170 \
-b "mds-vo-name=local,o=grid" -x 'objectclass=GlueSATop' \
GlueChunkKey GlueSAAccessControlBaseRule | grep -B 4 'GlueSAAccessControlBaseRule: atlas'
```

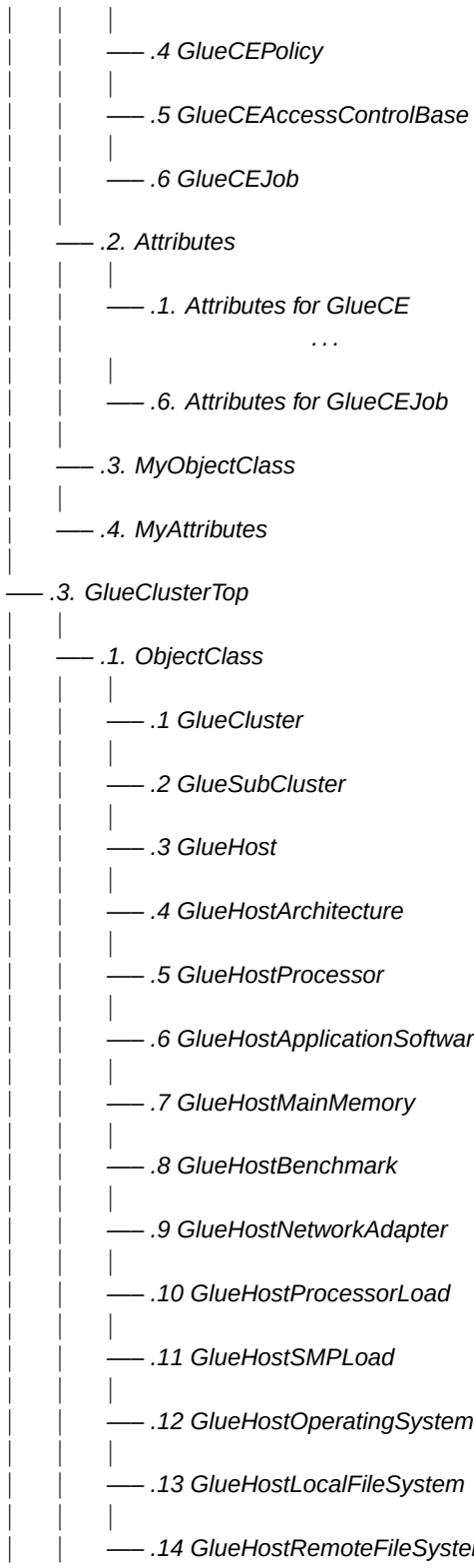
APPENDIX A THE GLUE SCHEMA

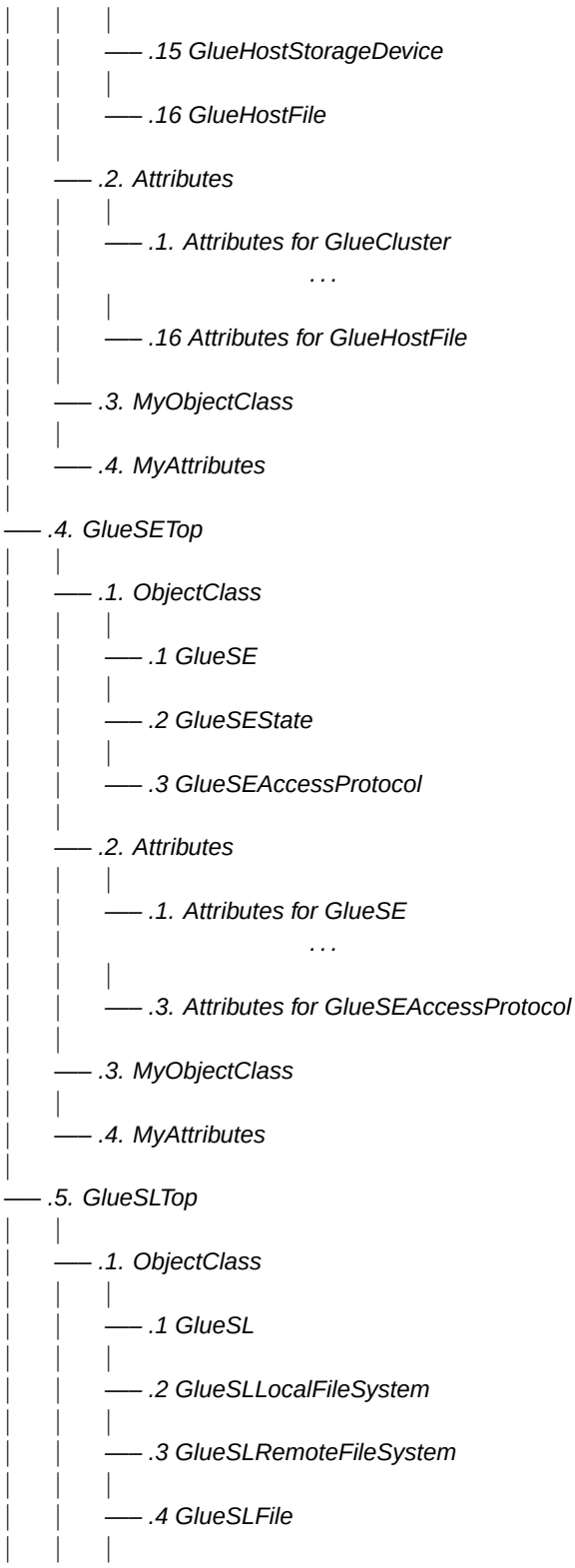
As explained earlier, the GLUE Schema describes what data about the elements in the Grid is stored for its use by the Information System.

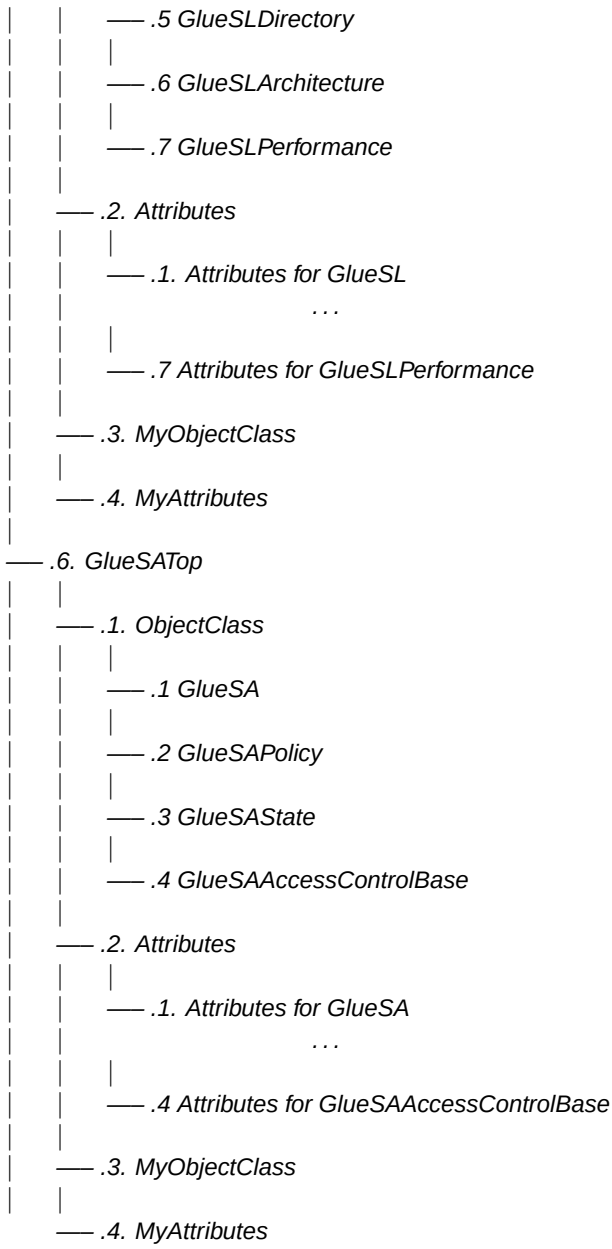
In this section, all the objectclasses of the LDAP hierarchy tree for the GLUE schema are described. First of all, the tree itself is showed. Then, the attributes for each one of the objectclasses (where the dynamique data is actually stored) are presented. The objectclasses are grouped in CE attributes, SE attributes and CE-SE binding attributes. Some of the attributes may actually be empty, even if they are defined in the schema.

A.1. THE GLUE SCHEMA LDAP TREE




.4 GlueCEPolicy
.5 GlueCEAccessControlBase
.6 GlueCEJob
.2. Attributes
.1. Attributes for GlueCE
. . .
.6. Attributes for GlueCEJob
.3. MyObjectClass
.4. MyAttributes
.3. GlueClusterTop
.1. ObjectClass
.1 GlueCluster
.2 GlueSubCluster
.3 GlueHost
.4 GlueHostArchitecture
.5 GlueHostProcessor
.6 GlueHostApplicationSoftware
.7 GlueHostMainMemory
.8 GlueHostBenchmark
.9 GlueHostNetworkAdapter
.10 GlueHostProcessorLoad
.11 GlueHostSMPLoad
.12 GlueHostOperatingSystem
.13 GlueHostLocalFileSystem
.14 GlueHostRemoteFileSystem


.15 GlueHostStorageDevice
.16 GlueHostFile
.2. Attributes
.1. Attributes for GlueCluster
. . .
.16 Attributes for GlueHostFile
.3. MyObjectClass
.4. MyAttributes
.4. GlueSETop
.1. ObjectClass
.1 GlueSE
.2 GlueSEState
.3 GlueSEAccessProtocol
.2. Attributes
.1. Attributes for GlueSE
. . .
.3. Attributes for GlueSEAccessProtocol
.3. MyObjectClass
.4. MyAttributes
.5. GlueSLTop
.1. ObjectClass
.1 GlueSL
.2 GlueSLLocalFileSystem
.3 GlueSLRemoteFileSystem
.4 GlueSLFile



A.2. ATTRIBUTES FOR THE COMPUTING ELEMENT

- ***CE (objectclass GlueCE)***
  - *GlueCEUniqueID: unique identifier for the CE*
  - *GlueCEName: human-readable name of the service*
- ***Info (objectclass GlueCEInfo)***

  - *GlueCEInfoLRMSType: name of the local batch system*
  - *GlueCEInfoLRMSVersion: version of the local batch system*
  - *GlueCEInfoGRAMVersion: version of GRAM*
  - *GlueCEInfoHostName: fully qualified name of the host where the gatekeeper runs*
  - *GlueCEInfoGateKeeperPort: port number for the gatekeeper*
  - *GlueCEInfoTotalCPUs: number of CPUs in the cluster associated to the CE*

- ***State (objectclass GlueCEState)***
  - *GlueCEStateRunningJobs: number of running jobs*
  - *GlueCEStateWaitingJobs: number of jobs not running*
  - *GlueCEStateTotalJobs: total number of jobs (running + waiting)*
  - *GlueCEStateStatus: queue status: queueing (jobs are accepted but not run), production (jobs are accepted and run), closed (jobs are neither accepted nor run), draining (jobs are not accepted but those in the queue are run)*
  - *GlueCEStateWorstResponseTime: worst possible time between the submission of a job and the start of its execution*
  - *GlueCEStateEstimatedResponseTime: estimated time between the submission of a job and the start of its execution*
  - *GlueCEStateFreeCPUs: number of CPUs available to the scheduler*

- ***Policy (objectclass GlueCEPolicy)***
  - *GlueCEPolicyMaxWallClockTime: maximum wall clock time available to jobs submitted to the CE*
  - *GlueCEPolicyMaxCPUTime: maximum CPU time available to jobs submitted to the CE*
  - *GlueCEPolicyMaxTotalJobs: maximum allowed total number of jobs in the queue*
  - *GlueCEPolicyMaxRunningJobs: maximum allowed number of running jobs in the queue*
  - *GlueCEPolicyPriority: information about the service priority*

- ***Access control (objectclass GlueCEAccessControlBase)***
  - *GlueCEAccessControlBaseRule: a rule defining any access restrictions to the CE. Current semantic: VO = a VO name, DENY = an X.509 user subject*

- ***Job (currently not filled, the Logging and Bookkeeping service can provide this information) (objectclass GlueCEJob)***
  - *GlueCEJobLocalOwner: local user name of the job's owner*
  - *GlueCEJobGlobalOwner: GSI subject of the real job's owner*
  - *GlueCEJobLocalID: local job identifier*
  - *GlueCEJobGlobalId: global job identifier*
  - *GlueCEJobGlueCEJobStatus: job status: **SUBMITTED**, **WAITING**, **READY**, **SCHEDULED**, **RUNNING**, **ABORTED**, **DONE**, **CLEARED**, **CHECKPOINTED***
  - *GlueCEJobSchedulerSpecific: any scheduler specific information*

- ***Cluster (objectclass GlueCluster)***

  - *GlueClusterUniqueID: unique identifier for the cluster*
  - *GlueClusterName: human-readable name of the cluster*
- ***Subcluster (objectclass GlueSubCluster)***
  - *GlueSubClusterUniqueID: unique identifier for the subcluster*
  - *GlueSubClusterName: human-readable name of the subcluster*
- ***Host (objectclass GlueHost)***
  - *GlueHostUniqueId: unique identifier for the host*
  - *GlueHostName: human-readable name of the host*
- ***Architecture (objectclass GlueHostArchitecture)***
  - *GlueHostArchitecturePlatformType: platform description*
  - *GlueHostArchitectureSMPSize: number of CPUs*
- ***Processor (objectclass GlueHostProcessor)***
  - *GlueHostProcessorVendor: name of the CPU vendor*
  - *GlueHostProcessorModel: name of the CPU model*
  - *GlueHostProcessorVersion: version of the CPU*
  - *GlueHostProcessorOtherProcessorDescription: other description for the CPU*
  - *GlueHostProcessorClockSpeed: clock speed of the CPU*
  - *GlueHostProcessorInstructionSet: name of the instruction set architecture of the CPU*
  - *GlueHostProcessorGlueHostProcessorFeatures: list of optional features of the CPU*
  - *GlueHostProcessorCacheL1: size of the unified L1 cache*
  - *GlueHostProcessorCacheL1I: size of the instruction L1 cache*
  - *GlueHostProcessorCacheL1D: size of the data L1 cache*
  - *GlueHostProcessorCacheL2: size of the unified L2 cache*
- ***Application software (objectclass GlueHostApplicationSoftware)***
  - *GlueHostApplicationSoftwareRunTimeEnvironment: list of software installed on this host*
- ***Main memory (objectclass GlueHostMainMemory)***
  - *GlueHostMainMemoryRAMSize: physical RAM*
  - *GlueHostMainMemoryRAMAvailable: unallocated RAM*
  - *GlueHostMainMemoryVirtualSize: size of the configured virtual memory*
  - *GlueHostMainMemoryVirtualAvailable: available virtual memory*
- ***Benchmark (objectclass GlueHostBenchmark)***
  - *GlueHostBenchmarkSI00: SpecInt2000 benchmark*
  - *GlueHostBenchmarkSF00: SpecFloat2000 benchmark*
- ***Network adapter (objectclass GlueHostNetworkAdapter)***

  - *GlueHostNetworkAdapterName: name of the network card*
  - *GlueHostNetworkAdapterIPAddress: IP address of the network card*
  - *GlueHostNetworkAdapterMTU: the MTU size for the LAN to which the network card is attached*
  - *GlueHostNetworkAdapterOutboundIP: permission for outbound connectivity*
  - *GlueHostNetworkAdapterInboundIP: permission for inbound connectivity*

- ***Processor load (objectclass GlueHostProcessorLoad)***
  - *GlueHostProcessorLoadLast1Min: one-minute average processor availability for a single node*
  - *GlueHostProcessorLoadLast5Min: 5-minute average processor availability for a single node*
  - *GlueHostProcessorLoadLast15Min: 15-minute average processor availability for a single node*
- ***SMP load (objectclass GlueHostSMPLoad)***
  - *GlueHostSMPLoadLast1Min: one-minute average processor availability for a single node*
  - *GlueHostSMPLoadLast5Min: 5-minute average processor availability for a single node*
  - *GlueHostSMPLoadLast15Min: 15-minute average processor availability for a single node*
- ***Operating system (objectclass GlueHostOperatingSystem)***
  - *GlueHostOperatingSystemOSName: OS name*
  - *GlueHostOperatingSystemOSRelease: OS release*
  - *GlueHostOperatingSystemOSVersion: OS or kernel version*
- ***Local file system (objectclass GlueHostLocalFileSystem)***
  - *GlueHostLocalFileSystemRoot: path name or other information defining the root of the file system*
  - *GlueHostLocalFileSystemSize: size of the file system in bytes*
  - *GlueHostLocalFileSystemAvailableSpace: amount of free space in bytes*
  - *GlueHostLocalFileSystemReadOnly: true if the file system is read-only*
  - *GlueHostLocalFileSystemType: file system type*
  - *GlueHostLocalFileSystemName: the name for the file system*
  - *GlueHostLocalFileSystemClient: host unique id of clients allowed to remotely access this file system*
- ***Remote file system (objectclass GlueHostRemoteFileSystem)***
  - *GlueHostLRemoteFileSystemRoot: path name or other information defining the root of the file system*
  - *GlueHostRemoteFileSystemSize: size of the file system in bytes*
  - *GlueHostRemoteFileSystemAvailableSpace: amount of free space in bytes*
  - *GlueHostRemoteFileSystemReadOnly: true if the file system is read-only*
  - *GlueHostRemoteFileSystemType: file system type*
  - *GlueHostRemoteFileSystemName: the name for the file system*
  - *GlueHostRemoteFileSystemServer: host unique id of the server which provides access to the file system*
- ***Storage device (objectclass GlueHostStorageDevice)***

  - *GlueHostStorageDeviceName: name of the storage device*
  - *GlueHostStorageDeviceType: storage device type*
  - *GlueHostStorageDeviceTransferRate: maximum transfer rate for the device*
  - *GlueHostStorageDeviceSize: size of the device*
  - *GlueHostStorageDeviceAvailableSpace: amount of free space*

- ***File (objectclass GlueHostFile)***
  - *GlueHostFileName: name for the file*
  - *GlueHostFileSize: file size in bytes*
  - *GlueHostFileCreationDate: file creation date and time*
  - *GlueHostFileLastModified: date and time of the last modification of the file*
  - *GlueHostFileLastAccessed: date and time of the last access to the file*
  - *GlueHostFileLatency: time taken to access the file in seconds*
  - *GlueHostFileLifeTime: time for which the file will stay on the storage device*
  - *GlueHostFileOwner: name of the owner of the file*

A.3. ATTRIBUTES FOR THE STORAGE ELEMENT

- ***Storage Service (objectclass GlueSE)***
  - *GlueSEUniqueId: unique identifier of the storage service (URI)*
  - *GlueSEName: human-readable name for the service*
  - *GlueSEPort: port number that the service listens*
  - *GlueSEHostingSL: unique identifier of the storage library hosting the service*
- ***Storage Service State (objectclass GlueSEState)***
  - *GlueSEStateCurrentIOLoad: system load (for example, number of files in the queue)*
- ***Storage Service Access Protocol (objectclass GlueSEAccessProtocol)***
  - *GlueSEAccessProtocolType: protocol type to access or transfer files*
  - *GlueSEAccessProtocolPort: port number for the protocol*
  - *GlueSEAccessProtocolVersion: protocol version*
  - *GlueSEAccessProtocolAccessTime: time to access a file using this protocol*
  - *GlueSEAccessProtocolSupportedSecurity: security features supported by the protocol*
- ***Storage Library (objectclass GlueSL)***
  - *GlueSLName: human-readable name of the storage library*
  - *GlueSLUniqueId: unique identifier of the machine providing the storage service*
  - *GlueSLService: unique identifier for the provided storage service*
- ***Local File system (objectclass GlueSLLocalFileSystem)***

- *GlueSLLocalFileSystemRoot: path name (or other information) defining the root of the file system*
- *GlueSLLocalFileSystemName: name of the file system*
- *GlueSLLocalFileSystemType: file system type (e.g. NFS, AFS, etc.)*
- *GlueSLLocalFileSystemReadOnly: true is the file system is read-only*
- *GlueSLLocalFileSystemSize: total space assigned to this file system*
- *GlueSLLocalFileSystemAvailableSpace: total free space in this file system*
- *GlueSLLocalFileSystemClient: unique identifiers of clients allowed to access the file system remotely*
- *GlueSLLocalFileSystemServer: unique identifier of the server exporting this file system (only for remote file systems)*

- ***Remote File system (objectclass GlueSLRemoteFileSystem)***
  - *GlueSLRemoteFileSystemRoot: path name (or other information) defining the root of the file system*
  - *GlueSLRemoteFileSystemName: name of the file system*
  - *GlueSLRemoteFileSystemType: file system type (e.g. NFS, AFS, etc.)*
  - *GlueSLRemoteFileSystemReadOnly: true is the file system is read-only*
  - *GlueSLRemoteFileSystemSize: total space assigned to this file system*
  - *GlueSLRemoteFileSystemAvailableSpace: total free space in this file system*
  - *GlueSLRemoteFileSystemServer: unique identifier of the server exporting this file system*
- ***File Information (objectclass GlueSLFile)***
  - *GlueSLFileName: file name*
  - *GlueSLFileSize: file size*
  - *GlueSLFileCreationDate: file creation date and time*
  - *GlueSLFileLastModified: date and time of the last modification of the file*
  - *GlueSLFileLastAccessed: date and time of the last access to the file*
  - *GlueSLFileLatency: time needed to access the file*
  - *GlueSLFileLifeTime: file lifetime*
  - *GlueSLFilePath: file path*
- ***Directory Information (objectclass GlueSLDirectory)***
  - *GlueSLDirectoryName: directory name*
  - *GlueSLDirectorySize: directory size*
  - *GlueSLDirectoryCreationDate: directory creation date and time*
  - *GlueSLDirectoryLastModified: date and time of the last modification of the directory*
  - *GlueSLDirectoryLastAccessed: date and time of the last access to the directory*
  - *GlueSLDirectoryLatency: time needed to access the directory*
  - *GlueSLDirectoryLifeTime: directory lifetime*
  - *GlueSLDirectoryPath: directory path*
- ***Architecture (objectclass GlueSLArchitecture)***

  - *GlueSLArchitectureType: type of storage hardware (i.e. disk, RAID array, tape library, etc.)*

- ***Performance (objectclass GlueSLPerformance)***
  - *GlueSLPerformanceMaxIOCapacity: maximum bandwidth between the service and the network*
- ***Storage Space (objectclass GlueSA)***
  - *GlueSARoot: pathname of the directory containing the files of the storage space*
- ***Policy (objectclass GlueSAPolicy)***
  - *GlueSAPolicyMaxFileSize: maximum file size*
  - *GlueSAPolicyMinFileSize: minimum file size*
  - *GlueSAPolicyMaxData: maximum allowed amount of data that a single job can store*
  - *GlueSAPolicyMaxNumFiles: maximum allowed number of files that a single job can store*
  - *GlueSAPolicyMaxPinDuration: maximum allowed lifetime for non-permanent files*
  - *GlueSAPolicyQuota: total available space*
  - *GlueSAPolicyFileLifeTime: lifetime policy for the contained files*
- ***State (objectclass GlueSAState)***
  - *GlueSAStateAvailableSpace: total space available in the storage space (in kilobytes)*
  - *GlueSAStateUsedSpace: used space in the storage space (in kilobytes)*
- ***Access Control Base (objectclass GlueSAAccessControlBase)***
  - *GlueSAAccessControlBase Rule: list of the access control rules*

A.4. ATTRIBUTES FOR THE CE-SE BINDING

The CE-SE binding schema represents a mean for advertising relationships between a CE and a SE (or several SEs). This is defined by site administrators and is used when scheduling jobs that must access input files or create output files from or to SEs.

- ***Associations between an CE and one or more SEs (objectclass GlueCESEBindGroup)***
  - *GlueCESEBindGroupCEUniqueID: unique ID for the CE*
  - *GlueCESEBindGroupSEUniqueID: unique ID for the SE*
- ***Associations between an SE and a CE (objectclass GlueCESEBind)***
  - *GlueCESEBindCEUniqueID: unique ID for the CE*
  - *GlueCESEBindCEAccesspoint: access point in the cluster from which CE can access a local SE*
  - *GlueCESEBindSEUniqueID: unique ID for the SE*

APPENDIX B THE GRID MIDDLEWARE

The Grid Middleware deployed in the LCG-1 service is reported below.

The operating system for the Computing Elements is Linux Red Hat 7.3, mainly running on IA32 computers.

*The LCG-2 Middleware layer uses components from **EDT (European DataTag)** 1.1, **EDG (European DataGrid)** 2.1 and VDT **(Virtual Data Toolkit)** 1.1.8. In the following we list the components from these packages/suites, which are currently used in LCG-2:*

- *EDG 2.1*
  - *EDG-WMS Workload Management System*
  - *Data Management System*
    - *EDG-RM (Replica Manager)*
    - *EDG-RLS (Replica Location Service)*
      - *Including the EDG-LRC (Local Replica Catalog)*
    - *EDG-RMC (Replica Metadata Catalog)*
  - *Fabric Management*
    - *EDG WP4 tools/procedures (LCFG, LCFG-Lite or manual procedures)*
    - *LCAS/LCMAPS (Local Centre Auth. System and Local Mapping)*
  - *Virtual Organization Management*
    - *EDG infrastructure and procedures*
  - *Information service*
    - *Information index BDII*
- *EDT 1.1*
  - *Monitoring system*
    - *Grid-ICE*
  - *Glue Schema LCG-EDT 1.1*
- *VDT 1.1.8*
  - *Core components:*
    - *Globus 2.2.4*
    - *Condor 6.4.7*
      - *Condor-G*
      - *ClassAds*
- *Information Service*
  - *Globus MDS*
    - *GRIS (Grid Resource Information Service)*
    - *GIIS (Grid Index Information Service)*

APPENDIX C JOB STATUS DEFINITION

As already mentioned in chapter 5, a job can find itself in one of several possible states, the definition of which is given in this table.

Status	***Definition***
SUBMITTED	*The job has been submitted by the user but not yet processed by the Network Server*
WAITING	*The job has been accepted by the Network Server but not yet processed by the Workload Manager*
READY	*The job has been assigned to a Computing Element but not yet transferred to it*
SCHEDULED	*The job is waiting in the Computing Element's queue*
RUNNING	*The job is running*
DONE	*The job has finished*
ABORTED	*The job has been aborted by the WMS (e.g. because it was too long, or the proxy certificated expired, etc.)*
CANCELLED	*The job has been cancelled by the user*
CLEARED	*The Output Sandbox has been transferred to the User Interface*

Only some transitions between states are allowed. These transitions are depicted in Figure 9.

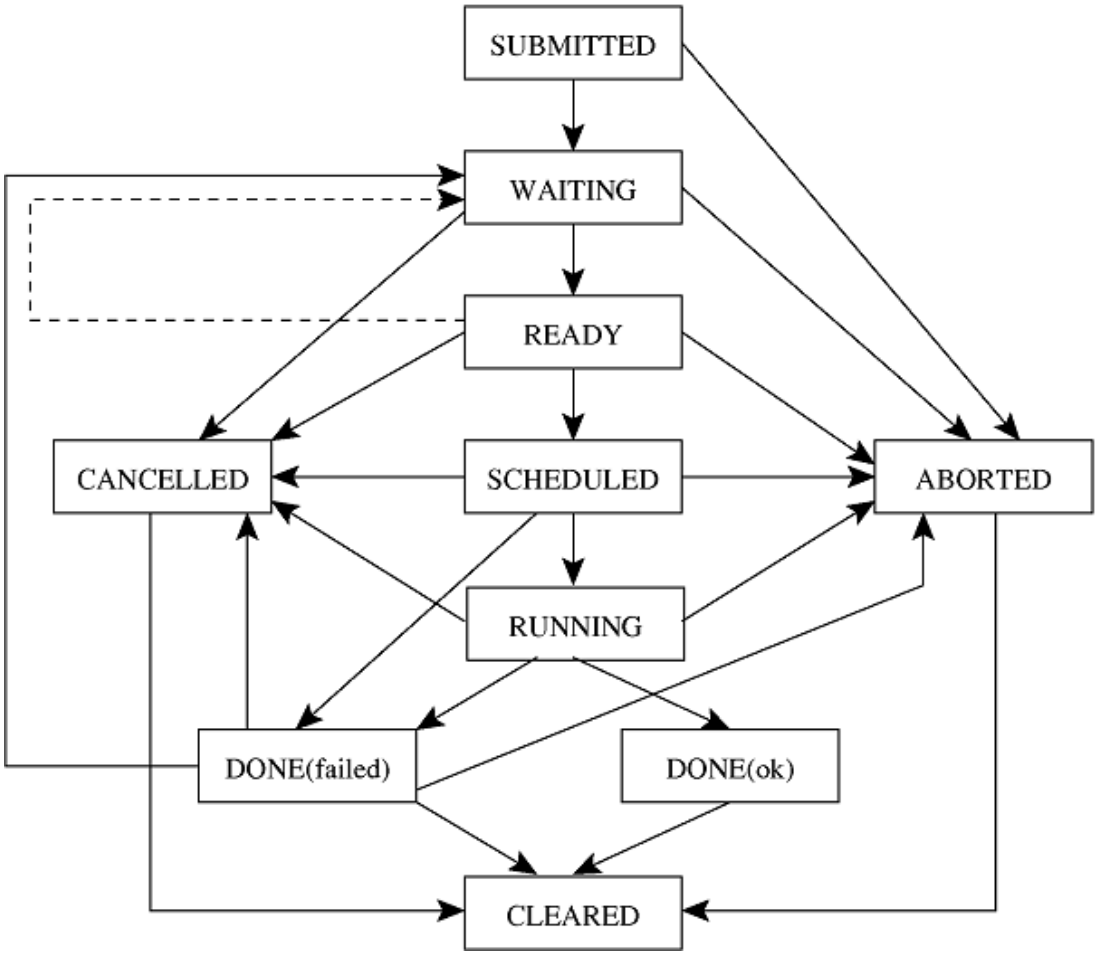


Figure 9: Possible job states in the LCG-2